HYBRID RECOMMENDER SYSTEM FOR PERSONALIZED PATIENT CARE

UNDERGRADUATE THESIS

Submitted to the International Undergraduate Program in Industrial Engineering in Partial Fulfilment of Requirement for the Degree of Sarjana Teknik at the Faculty of Industrial Technology Universitas Islam Indonesia

Name : Muh. Albara Husain Haq B.

Student Number : 18522360

UNDERGRADUATE PROGRAM IN INDUSTRIAL ENGINEERING

FACULTY OF INDUSTRIAL TECHNOLOGY

UNIVERSITAS ISLAM INDONESIA

YOGYAKARTA

2025

AUTHENTICITY STATEMENT

For the sake of Allah SWT, I admit this work is the result of my own work, except for the excerpts and summaries from which I have explained the source. If in the future, it turns out that my confession is proven to be untrue and violates the legal regulations in the paper and intellectual property rights. In that case, I am willing to get a diploma that I have received to be withdrawn by Universitas Islam Indonesia.

Yogyakarta, *August 29- 2025*



Muh. Albara
Husain Haq B.

18522360

SUPERVISOR APPROVAL SHEET

HYBRID RECOMMENDER SYSTEM FOR PERSONALIZED PATIENT CARE

UNDERGRADUATE THESIS

Written by:

Name : Muh. Albara Husain Haq B.
Student Number : 18522360

Yogyakarta, August 29, 2025

Supervisor,



(Rohmatul Fajriyah, Dr.techn., S.Si., M.Si)

EXAMINERS' APPROVAL PAGE

HYBRID RECOMMENDER SYSTEM FOR PERSONALIZED PATIENT CARE

UNDERGRADUATE THESIS

Written by:

Name : Muh Albara Husain Haq B.

Student Number : 18 522 360

Has been defended before the board of examiners in partial fulfillment of the requirement for Bachelor Degree in Teknik Industri at the Faculty of Industrial Technology Universitas Islam Indonesia

Yogyakarta, ***August 29 2025***

Board of Examiners

Rohmatul Fajriyah, Dr.techn., S.Si., M.Si	
Chairman	
Ir. Muhammad Ridwan Andi Purnomo, S.T., M.Sc., Ph.D., IPM.	
Member I	
Dr. Agus Mansur, S.T., M.Eng.Sc	
Member II	

Acknowledged by,
Head of Undergraduate Program in Industrial Engineering
Faculty of Industrial Technology
Universitas Islam Indonesia

Ir. Muhammad Ridwan Andi Purnomo, S.T., M.Sc., Ph.D., IPM

PREFACE

Assalamuálaikum Warahmatullahi Wabarakatuh,

Allhamdulilah, all praise to Allah SWT, because only with his permission the author can finish the and also write this undergraduate thesis. Shalawat and greetings to the prophet Muhammad SAW who has to save mankind from the jahiliyyah era to the Islamiyah era and also give syafaat in yaumul akhir.

This report was made to fulfill one of the requirements for completing a degree in Industrial Engineering at Universitas Islam Indonesia. The author realizes if he can't finish the project without the help, the author would like to thank:

1. The author's beloved parents Hadi Rohman and Murtiningsih also for my big families who always give support, motivation, and prays.
2. Mr. Muhammad Ridwan Andi Purnomo, S.T., M.Sc., Ph.D. as the Head of Undergraduate Program Department of Industrial Engineering Universitas Islam Indonesia.
3. Danang Setiawan, S.T., M.T. as the Secretary of Undergraduate Program Department of Industrial Engineering – International Program Universitas Islam Indonesia.
4. Rohmatul Fajriyah, Dr.techn., S.Si., M.Si. as the supervisor who always provides guidance and knowledge to help the author finish this internship report.
5. Ms. Devy Nurrahmah, S.Kom Staff of Undergraduate Program Department of Industrial Engineering – International Program Universitas Islam Indonesia.
6. All IP Students batch 2018.

Wassalamuálaikum Warahmatullahi Wabarakatuh.

Sleman, 29-August-2025

Muh. Albara Husain Haq B

NIM 18522360

ABSTRACT

This study presents the development of a hybrid recommender system designed to support clinical decision-making in healthcare. The system integrates four distinct approaches—collaborative filtering with matrix factorization, neural collaborative filtering, content-based filtering, and knowledge-based filtering—within a unified neural network framework. By leveraging both patient-specific data and domain knowledge, the system provides personalized recommendations for medical procedures. Experimental results on the MIMIC-IV dataset demonstrate the effectiveness of the proposed model, achieving a Mean Absolute Error (MAE) of 0.0842 and a Mean Absolute Percentage Error (MAPE) of 8.42% on the test set. These outcomes highlight the huge upside potential to be applied such as tailoring services to patient profiles, optimizing resource allocation, and supporting targeted healthcare promotion. These results demonstrate the practical value of hybrid recommendation models in bridging data-driven insights with medical domain knowledge.

Keywords: Hybrid Recommender System, Clinical Decision Support, Personalized Healthcare, Collaborative Filtering, MIMIC-IV Dataset, Healthcare

Table of Content

1 AUTHENTICITY STATEMENT i
2 SUPERVISOR APPROVAL SHEET ii
3 EXAMINERS' APPROVAL PAGE iii
4 PREFACE iv
5 ABSTRACT v
Table of Content vi
IMAGE LIST ix
IMAGE LIST ix
Chapter I 1
Introduction 1
1.1 Background information on clinical decision support systems and recommender systems 1
1.2 Problem formulation 3
1.3 Problem Limitation 3
1.4 Type of Research and Method of Analysis 4
1.5 Research objectives 4
1.6 Research benefits 4
Chapter II 5
Literature Review 5
2.1 Recommender systems 5
2.1.1. Content-based Filtering 6
2.1.2. Collaborative filtering 6
2.1.3. Matrix factorization 6
2.1.4. Knowledge-based filtering 7
2.2 Hybrid recommender system 7
2.3 Deep learning 8
2.4 Optimization Techniques, Metrics, and Activation Functions 9
3.4.1. Activation function 9
3.4.2. Optimization algorithms. 10
3.4.3. Training metrics 11
3.4.4. Testing metrics 12
2.5 State-of-the-art of recommender systems in healthcare 12

Chapter III....14
Research Methodology....14
3.1 Research Flow....14
3.2 Business understanding....15
3.3 Data collection....16
3.4 Data Reduction and Preparation....18
3.4.1 User DataFrame....19
3.4.2 Procedures DataFrame....19
3.4.3 Context DataFrame....20
3.5 Data exploration....21
3.6 Data cleaning....22
3.7 Data standardization....22
3.8 Data splitting for training and testing....22
3.9 Design recommendation algorithms....23
3.9.1 Collaborative filtering using matrix factorization.....24
3.9.2 Collaborative filtering using a neural network....24
3.9.3 Content-based filtering using a neural network....25
3.9.4 Knowledge-based neural network....26
3.10 Combine the four algorithms....27
3.11 Compiled with optimizer and loss function....28
3.12 Set the hyperparameter....28
3.13 Train....28
3.14 Evaluate....29
3.14.1 Training....29
3.14.2 Matrics evaluation....29
3.15 Generate 10 recommendation....29
Chapter IV....30
Results and Discussion....30
4.1 Business Understanding....30
4.2 Data collection....30
4.3 Data reduction and preparation....34
4.2.1 User Dataframe....38
4.2.2 Product/procedure Dataframe....38
4.2.3 Context Dataframe....39

4.3 Data exploration 39
4.3.1 Data Distribution 39
4.3.2 Correlation Between Features and Products 40
4.3.3 Plotting Procedures Features 40
4.3.4 Plotting Sequence Number and Anchor Ages 41
4.4 Data cleaning 42
4.5 Data Standardization 43
4.6 Design Artificial Neural Network 44
4.7 Combine the four algorithms 47
4.8 Compiled with the optimizier, loss function and metrics 47
4.9 Set the hyperparameter on the model 48
4.10 Train Artificial Neural Network 48
4.11 Generate 10 recommendation 49
Chapter V 52
System Testing and Discussion 52
5.1 Presentation and discussion of the system testing results 52
5.1.1 Training history 52
5.1.2 Evaluation metrics 56
5.1.2.1 MAE 56
5.1.2.2 MAPE 56
5.2 Generate 10 procedures 57
Chapter VI 59
Conclusion and Future Work 59
6.1 Summary of the study and main findings 59
6.2 Future recommendations for improving the hybrid recommender system and its potential applications 60
6.3 Conclusion and final remarks 60
Reference: 62
Appendix: Additional Technical Details 64

IMAGE LIST

Figure 1 Diagram of key interactions in knowledge-based and non-knowledge-based CDSS. (Sutton et al., 2020) 2
Figure 2 Parallel and sequential ensemble (Aggarwal, 2016) 8
Figure 3 Adam optimizier (Kingma & Ba, 2014) 11
Figure 4 research flowchart 15
Figure 5 Data collection flowchart 17
Figure 6 Data Collection 17
Figure 7 Dara reduction and preparation flowchart 18
Figure 8 Data Cleaning Flowchart 22
Figure 9 spliting data 22
Figure 10 Design recommendation algorithms Flowchart 23
Figure 11 Neural Collaborative filtering framework(He et al., 2017) 25
Figure 12 Master Data 31
Figure 13 procedures master data 32
Figure 14 diagnose master data 33
Figure 15 Y or seq_num distribution Master Data 34
Figure 16 Procedures after reduction 36
Figure 17 Diagnose after reduction 36
Figure 18 y distribution after reduction 36
Figure 19 Complete data after preparation 37
Figure 20 User dataframe 38
Figure 21 Procedures features 38
Figure 22 Procedures dataframe 38
Figure 23 Context dataframe 39
Figure 24 Context Features 39
Figure 25 y distribution toward seq_num 40
Figure 26 Procedures towards Featues correlation 40
Figure 27 Features distribution in each patient 40
Figure 28 Features Distribution frequency 41
Figure 29 the distribution of anchor_age for a unique value of seq_num 42
Figure 30 y distribution after data cleaning. 43
Figure 31 training history plot 56

TABLE LIST

Table 1 The shape of the 'master_data' dataframe 31
Table 2 Reduced master data shape 35

Chapter I

Introduction

1.1 Background information on clinical decision support systems and recommender systems

Accurate decision making in healthcare service is a key part of a physician's job. This practice involves understanding the patient's medical history, performing a thorough physical examination, ordering and interpreting diagnostic tests, and using clinical guidelines and evidence-based medicine to identify the most appropriate treatment options (Croskerry, 2009). It also requires good communication skills to explain the treatment plan to the patient and coordinate care with other healthcare providers. Making accurate treatment decisions can be challenging, especially in complex or emergencies, and requires a combination of knowledge, experience, and judgment (Gräßer et al., 2021).

Clinical decision support systems (CDSS) have been used in healthcare for several decades, with the first systems being developed in the 1980s. CDSSs have evolved and now use various algorithms and data analytics to provide healthcare professionals with relevant information and recommendations. They can be utilized in different clinical settings, such as hospitals, clinics, and primary care practices, to assist with a range of clinical decisions, including diagnosis, treatment, and medication management. CDSSs can be classified into two broad categories: knowledge-based and non-knowledge-based systems(Sutton et al., 2020). Knowledge-based systems are structured using if-then rules, built on top of a knowledge base, while non-knowledge-based systems employ machine learning and artificial intelligence. See Figure 1.

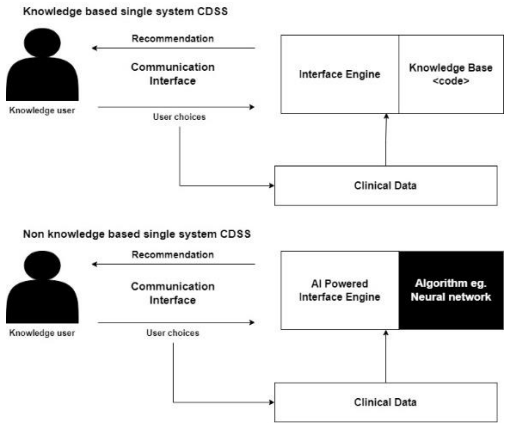


Figure 1 Diagram of key interactions in knowledge-based and non-knowledge-based CDSS. (Sutton et al., 2020)

CDSSs could improve the quality of clinical decisions and healthcare processes. CDSS can improve the accuracy and reliability of clinical decisions, reduce errors and adverse outcomes, and improve patient outcomes. However, there are also challenges associated with CDSS, including the need for significant time and resources to set up and maintain the systems and concerns about the accuracy and reliability of the recommendations provided (Jones et al., 2021).

Traditional CDSSs are typically based on pre-defined rules (IF-THEN statements) that are created based on literature-based, practice-based, or patient-directed evidence to provide recommendations that are designed to provide recommendations based on specific criteria. However, these rules and algorithms may not consider each patient's unique circumstances and individual needs. Recently, a focus has been on developing artificial intelligence. Artificial Intelligence has the potential to overcome some of the limitations of traditional CDSS and provide more personalized recommendations to support clinical decision-making.

To address these challenges, we have developed a hybrid recommender system that combines four recommendation algorithms: collaborative filtering using matrix factorization, collaborative filtering using a neural network, content-based filtering using product features, and knowledge-based filtering using context. Our hybrid system can provide accurate recommendations to support clinical decision-making by healthcare professionals, hospitals, and patients.

Our research aims to design and develop a hybrid recommender system that can provide personalized clinical recommendations. By combining multiple recommendation algorithms, our system intends to overcome the limitations of

traditional CDSS and provide accurate recommendations that we measure with accuracy metrics: Mean Absolute error and Mean percentage error. We believe that this system has the potential to improve the quality of patient care, reduce the risk of errors and adverse outcomes, and enhance hospitals' performance, resulting in higher patient satisfaction.

1.2 Problem formulation

How to develop a hybrid recommender system for clinical decision support combining different recommendation algorithms to improve hospital performance by providing more accurate and reliable recommendations that benefit doctors, hospitals, and patients?

1.3 Problem Limitation

Limitations of the problem in this writing are as follows:

1. Data availability and quality: The effectiveness of a recommendation system is highly dependent on the quality and quantity of data available for training. We massively reduce our data, and it can lead to flawed recommendations. We reduce because of hardware limitations.
2. User acceptance: Recommendation systems may not always be accepted by users, especially if they feel that the recommendations are not relevant or appropriate. It becomes our limitation since we get this data from a public resource.
3. Legal and ethical considerations: There are legal and ethical considerations to consider when using recommendation systems in the healthcare context, such as privacy, informed consent, and the potential for discrimination.
4. Limited generalizability: The results of a recommendation system may not be generalizable to other populations or settings and may not always be applicable in different clinical contexts
5. Dependence on technology: A recommendation system is only as good as the technology implemented. If the technology fails or is not updated regularly, the recommendation system may not function properly.

1.4 Type of Research and Method of Analysis

This final project is included in the machine learning and deep learning method category for implementing CDSS design & development using the Hybrid Recommender system.

1.5 Research objectives

Despite the challenge of accurate and reliable ethical issues toward recommendations to support clinical decision-making in the healthcare system, there is still a challenge. We aim to explore the potential benefits of a hybrid recommender system for clinical decision support that combines different recommendation algorithms to improve hospital performance and benefit doctors, hospitals, and patients.

1.6 Research benefits

Through this research, we hope to contribute to the field of clinical decision support and provide a helpful tool that can help :

1. Hospitals to increase their performance in healthcare delivery
2. Healthcare professionals make more informed and accurate decisions about patient care.
3. Other physicians, not only doctors but the administrative staff and nurses get recommendations related to their knowledge and task.
4. Hospitals to personalized advertisement offerings to patients, helping to engage them in their treatment decisions and improve patient satisfaction
5. Help patients to understand their treatment options and make informed decisions about their care.

Chapter II

Literature Review

2.1Recommender systems

A recommender system is an information filtering system that predicts and recommends items to users based on their preferences, historical behavior, and other relevant data. These systems are widely used in e-commerce, social media, and other online platforms to provide personalized recommendations to users. The basic principle of recommendations is that significant dependencies exist between user and item-centric activity(Aggarwal, 2016a).

The development of recommender systems can be traced back to the early 1990s when researchers started exploring collaborative filtering techniques. Collaborative filtering is a technique that uses the behavior of similar users to make recommendations. The GroupLens research group developed the first recommender system at the University of Minnesota in 1992. This system was used to recommend movies to users based on their ratings of other movies(Aggarwal, 2016a).

One of the most well-known success stories of a recommender system is the recommendation engine used by Amazon.com. Based on collaborative filtering, amazon's recommender system suggests products based on user data like purchase and browsing history, ratings, and search queries. It collects customer-specific data through authentication and uses a 5-point rating scale with explanations for recommendations. Implicit and explicit feedback are used, and the system has been highly successful in increasing sales and customer satisfaction(Aggarwal, 2016a).

Another success story is the recommendation system used by Netflix to recommend movies and TV shows to its subscribers. Netflix's system used to provide personalized recommendations to users. Netflix also created the Netflix Prize contest, which provided a platform for contestants to submit collaborative filtering algorithms to predict movie ratings. The contest used a data set of over 100 million movie ratings and offered prizes for improving the accuracy of Netflix's recommendation algorithm, Cinematch. The contest popularized many recommendation algorithms, including latent factor models. The contest's evaluation design was well-executed to prevent overfitting, and it contributed significantly to recommendation research(Aggarwal, 2016a).

The goal of a recommender system is to suggest relevant items to users based on their preferences and behavior. The recommender system achieves this by carefully selecting items to recommend to users, which in turn increases the sales volume and profits for the merchant (Aggarwal, 2016a). Hybrid recommender systems combine multiple recommendation techniques to improve recommendation accuracy and diversity and bring this recommendation into customer attention. This literature review will discuss state-of-the-art research on hybrid recommender systems.

Recommender systems are algorithms that provide personalized recommendations to users. These systems are widely used in e-commerce, entertainment, and social media industries to improve user engagement and satisfaction. Recommender systems work as computer programs that use data to analyze user data and provide personalized recommendations. These recommendations are based on various factors, such as user behavior, preferences, ratings, and social interactions.

There are several main types of recommender systems: content-based, collaborative, and hybrid. Each type of recommender system has its advantages and limitations.

2.1.1. Content-based Filtering

Content-based recommender systems recommend items to users based on the attributes of items they interacted. They can provide accurate recommendations for new items but may lack diversity and effectiveness for new users. They use training data to create a user-specific model that predicts whether the user will like an item. Content-based methods can also use user-specified keywords to match with item descriptions for recommendations, which is helpful in cold-start scenarios(Aggarwal, 2016a).

One advantage of content-based recommenders is that they can provide highly accurate recommendations when there is a lot of data available on the attributes of the items. One limitation of these systems is that they may have difficulty providing diverse recommendations, as they tend to focus on the attributes most similar to the items the user has already interacted.

2.1.2. Collaborative filtering

Collaborative filtering is a recommender system that recommends items to users based on the behavior of similar users. For instance, an online retailer may suggest products to a user based on the purchasing history of other users who have bought similar items(Aggarwal, 2016a).

One advantage of collaborative filtering is that it can create highly personalized recommendations by leveraging the behavior of users with similar interests. The limitation of collaborative filtering is that it may struggle to provide recommendations when there is not enough data available on user behavior. This can lead to inaccurate or irrelevant recommendations, especially for new users or niche items that are not frequently purchased.

2.1.3. Matrix factorization

Matrix factorization is a technique used in recommender systems that involves breaking down a large matrix into smaller, simpler matrices in order to predict missing values. In the context of a recommendation system, the large matrix represents the users and items, and the entries in the matrix represent the ratings or preferences of users for particular items. Matrix factorization can be used to discover latent factors that influence user-item interactions and then use them to make personalized recommendations(Aggarwal, 2016d).

The process of matrix factorization involves finding two lower-dimensional matrices that when multiplied together, approximate the original matrix. The two matrices are referred to as the user and item latent factors. The user latent factor captures the characteristics of the users, while the item latent factor captures the attributes of the items. By multiplying the user and item latent factors together, the original matrix can be reconstructed and missing entries can be predicted.

2.1.4. Knowledge-based filtering

Knowledge-based recommender system suggests items to users based on domain knowledge. Knowledge-based recommender systems are useful when there is a lot of domain knowledge available, particularly for items not chosen often(Aggarwal, 2016c).

Knowledge-based accuracy is highly dependent on the accuracy and completeness of the underlying knowledge base make this to become the limitation of these systems. Inaccurate or outdated information can lead to incorrect recommendations, which can be especially problematic in domains where new knowledge is constantly generated, such as medicine or technology.

2.2 Hybrid recommender system

Hybrid recommender systems combine multiple recommendation techniques to give users more accurate and personalized recommendations. These systems can overcome some of the limitations of traditional recommender systems by combining different algorithms. Hybrid recommender systems offer several advantages over traditional recommender systems, including:

1. Improved Accuracy: By combining multiple recommendation techniques, hybrid systems can provide users with more accurate and personalized recommendations.
2. Increased Coverage: Hybrid systems can provide recommendations for a broader range of items and users by leveraging different recommendation techniques.
3. Flexibility: Hybrid systems can be adapted to different data types and user preferences.

The need for multiple algorithms and data integration can make the hybrid technique more complex and costly to develop and maintain than traditional systems. They can also be more challenging to interpret and explain to users, leading to a lack of transparency and trust in the recommendations provided.

There are three primary design approaches to creating a hybrid recommender system: ensemble design, monolithic design, and mixed systems. Ensemble design

involves combining the results of multiple off-the-shelf algorithms to create a more robust output. Monolithic design involves creating an integrated recommendation algorithm using various data types, while mixed systems present recommendations from multiple algorithms side by side. The choice of design depends on the specific recommendation problem and the availability of data(Aggarwal, 2016b).

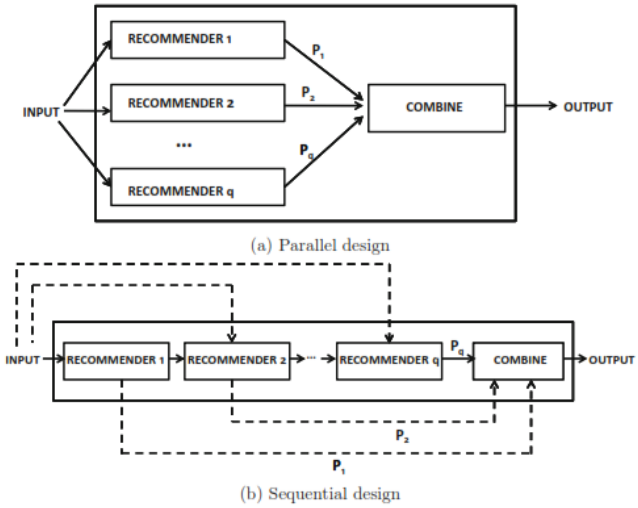


Figure 2 Parallel and sequential ensemble (Aggarwal, 2016)

2.3 Deep learning

Deep learning is a subfield of machine learning that trains computers to learn from data and improve their performance on a specific task. It mimics the way humans learn by creating a hierarchy of concepts or representations. This approach avoids the need for humans to specify all of the knowledge that the computer needs. The hierarchy allows the computer to learn complex concepts by building them out of simpler ones. For example, we might have three functions $f^{(1)}$, $f^{(2)}$, and $f^{(3)}$ connected in a chain, to form $f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x)))$. These chain structures are the most commonly used structures of neural networks. In this case, $f^{(1)}$ is called the first layer of the network, $f^{(2)}$ is called the second layer, and so on. The overall length of the chain gives the depth of the model. For this reason, this approach called deep learning(Ian Goodfellow et al., 2016).

A hybrid recommender system that combines deep learning with content-based filtering, collaborative filtering, and matrix factorization can provide highly accurate and personalized recommendations. Deep learning algorithms can be used to learn the latent factors that influence user-item interactions. In contrast, content-based filtering can be used to recommend items based on their attributes, and collaborative filtering can be used to recommend items based on similar user behavior. Matrix factorization can also uncover hidden relationships between users and items. By combining these techniques, a hybrid recommender system can leverage the strengths of each approach to provide more accurate and diverse recommendations.

Combining multiple recommendation algorithms in a hybrid recommendation system is possible due to the flexibility and adaptability of neural networks. Neural networks are machine learning algorithms that can learn and adapt to new data and patterns by adjusting the weights and biases of the connections between their nodes(cakir et al., 2020). This allows neural networks to learn and make predictions or recommendations based on complex and diverse data.

1. Collaborative Filtering (CF) - This part uses matrix factorization and neural network techniques to learn user and product embeddings and calculate their similarity.
2. Content-Based Filtering (CBF) - This part takes the product features as input and uses a dense layer to learn the features' importance.
3. Knowledge-Based Filtering (KBF) - This part takes the context variables as input and uses a dense layer to learn the context's importance.

2.4 Optimization Techniques, Metrics, and Activation Functions

Optimization techniques, metrics, and activation functions are important components of a deep-learning neural network. Choose appropriate optimization techniques, metrics, and activation functions by experimenting with different combinations of these components can help to improve the performance of the neural network

3.4.1. Activation function

Activation functions are mathematical functions that are applied to the output of a layer in a neural network to introduce non-linearity. Non-linearity is important for neural networks to learn complex relationships in data, which would not be possible with linear functions. The activation function typically applies a threshold or sum of the weighted inputs, determining whether the neuron produces an output signal. The output of a neuron is then fed as input to other neurons in the network, creating a hierarchy of layers. By applying activation functions, neural networks can learn and model more complex patterns and relationships in the data(Aggarwal, 2018). In the next chapter, we will use several activation functions in our deep-learning architecture

- ReLU (Rectified Linear Unit)
  Returns the input if it is positive and zero otherwise.

$$f(x) = \max(0, x) \tag{1}$$

- Linear.
  A linear activation function that returns the input as it is.

$$f(x) = x \tag{2}$$

3.4.2. Optimization algorithms.

Optimization algorithms are used to update the weights and biases of a neural network during the training process to minimize the loss function. The loss function measures the difference between the predicted output of the neural network and the actual output. Optimization algorithms enable us to continue updating model parameters and minimize the value of the loss function as evaluated on the training set(Ian Goodfellow, 2016).

In the upcoming chapter, we will be utilizing the Adaptive Moment Estimation optimizer (Kingma & Ba, 2014) in our deep-learning architecture. This is popular optimization algorithm that computes adaptive learning rates for each parameter. The Adam optimizer updates the weights and biases of a neural network using the following equation:

$$w(t+1) = w(t) - \alpha * m(t) / (\text{sqrt}(v(t)) + \text{epsilon}) \quad (3)$$

$$b(t+1) = b(t) - \alpha * m(t) / (\text{sqrt}(v(t)) + \text{epsilon}) \quad (4)$$

Where:

- w(t) and b(t) are the weights and biases at time t
- alpha is the learning rate
- m(t) and v(t) are the first and second moment estimates of the gradient, respectively
- epsilon is a small constant added to prevent division by zero

Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t.

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
$m_0 \leftarrow 0$ (Initialize 1st moment vector)
$v_0 \leftarrow 0$ (Initialize 2nd moment vector)
$t \leftarrow 0$ (Initialize timestep)
while θ_t not converged **do**
$t \leftarrow t + 1$
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
$\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
$\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
end while
return θ_t (Resulting parameters)

Figure 3 Adam optimizier (Kingma & Ba, 2014)

3.4.3. Training metrics

Measures the ability of the algorithm to predict user preferences based on historical data during the training phase.

- Loss function

A loss function is used to optimize the model during training by measuring the error between the predicted and true values of the target variable. The optimizer will use the calculated loss value to update the model weights in order to minimize the loss function.

The Loss function is a measure of the difference between the predicted values and the actual values on the training data. It indicates how well the algorithm is able to fit the training data. A lower value of the Training Loss indicates that the algorithm is making better predictions on the training data. We use mean absolute error as the loss function, which means that the reported loss value is the mean absolute error on the training data.

The mean absolute error (MAE) loss function can be expressed as:

$$MAE = 1/n * \Sigma |y - y_pred| \quad (5)$$

Where:

- n: number of samples
- y: true value
- y_pred: predicted value

- Metrics

Metrics in the model during training phase measuring the error between the predicted and true values of the target variable. provides an additional evaluation of the model's performance during training and can be useful for comparing different models or tuning hyperparameters. Unlike the loss function, the metric does not directly influence the model's training process or the model's optimization .

The mean absolute percentage error (MAPE) metric can be expressed as:

$$MAPE = 1/n * \Sigma (|y - y_pred|/y) \quad (6)$$

Where:

- n: number of samples
- y: true value
- y_pred: predicted value

3.4.4. Testing metrics

Evaluation Testing metrics for recommender systems measure the algorithm's performance on a test data set independent of the training data. These metrics are essential to assess the algorithm's performance in the real world. In the next chapter we will use both MAE and MAPE as our testing metrics

2.5State-of-the-art of recommender systems in healthcare

Recommender Systems have gained significant interest in healthcare due to their potential to assist medical professionals and patients in decision-making, improve patient outcomes, and reduce healthcare costs. Here are state-of-the-art also examples of related work in this area:

Table 1 Related work

Case Study	Description	Approach
A Hybrid Recommender System for Patient-Doctor Matchmaking in Primary Care(Han et al., 2018)	The development and mechanism for matching patients with family doctors in primary care.	a hybrid recommender system to provide patients with a list of family doctor recommendations.
Diabetic Healthcare Recommendation System (Bankhele et al., 2017)	The recommender system helps with diabetes self-management	The recommendation is done using User based collaborative filtering, the system asks the user to enter a predetermined set of parameters which are matched with other

		patients parameters stored in the database, the database consists of past cases of patients who have been diagnosed with diabetes and treated
Disease Prediction and Prevention Using Recommender System (Nasiri et al., 2016)	offer appropriate treatment for chronic patients who are treated during time	The collaborative filtering approach uses other users' ratings to predict the target users' unknown ratings.
A Framework of Hybrid Recommender System for Personalized Clinical Prescription(Zhang et al., 2015)	help general practitioners cope with the challenges of clinical prescription and make informed decisions based on the mass of medical knowledge hidden in history medical records.	hybrid recommender system that integrates artificial neural networks and case-based reasoning to aid general practitioners in making clinical decisions.

Chapter III

Research Methodology

3.1Research Flow

We build a clinical decision recommendation system in this study and want to ensure that this study is transparent and easy to understand. We try to provide the logic of this study and make the reader understand how we obtained the results. Detailing the analysis method and the stages and processes involved in the study also allows other researchers to replicate the study, which is essential for verifying the results and increasing the credibility of the research.

In order to complete several stages and processes, we will show steps on how this study will be done :

1. Bussiness understanding
2. Collect data on patient characteristics, procedure characteristics, and clinical outcomes from a relevant dataset.
3. Reduce data and Prepare data
    a. User DataFrame
    b. Procedures DataFrame
    c. Context DataFrame
4. Explore the data
5. Clean the data
6. Preprocess by standardizing the data as needed.
7. Split the data into training and test sets with a predetermined split ratio.
8. Design recommendation algorithms:
    a. collaborative filtering using matrix factorization,
    b. collaborative filtering using a neural network,
    c. content-based filtering using neural networks, and
    d. knowledge-based filtering using neural network.
9. Combine the four algorithms into a hybrid system by concatenated and passed through a final dense layer with a linear activation function to produce the final recommendation
10. Compiled with the and the loss function
11. Set the hyperparameter on the model
12. Train the model.
13. Evaluate the performance of the model on the test set
14. Generate 10 recommendation

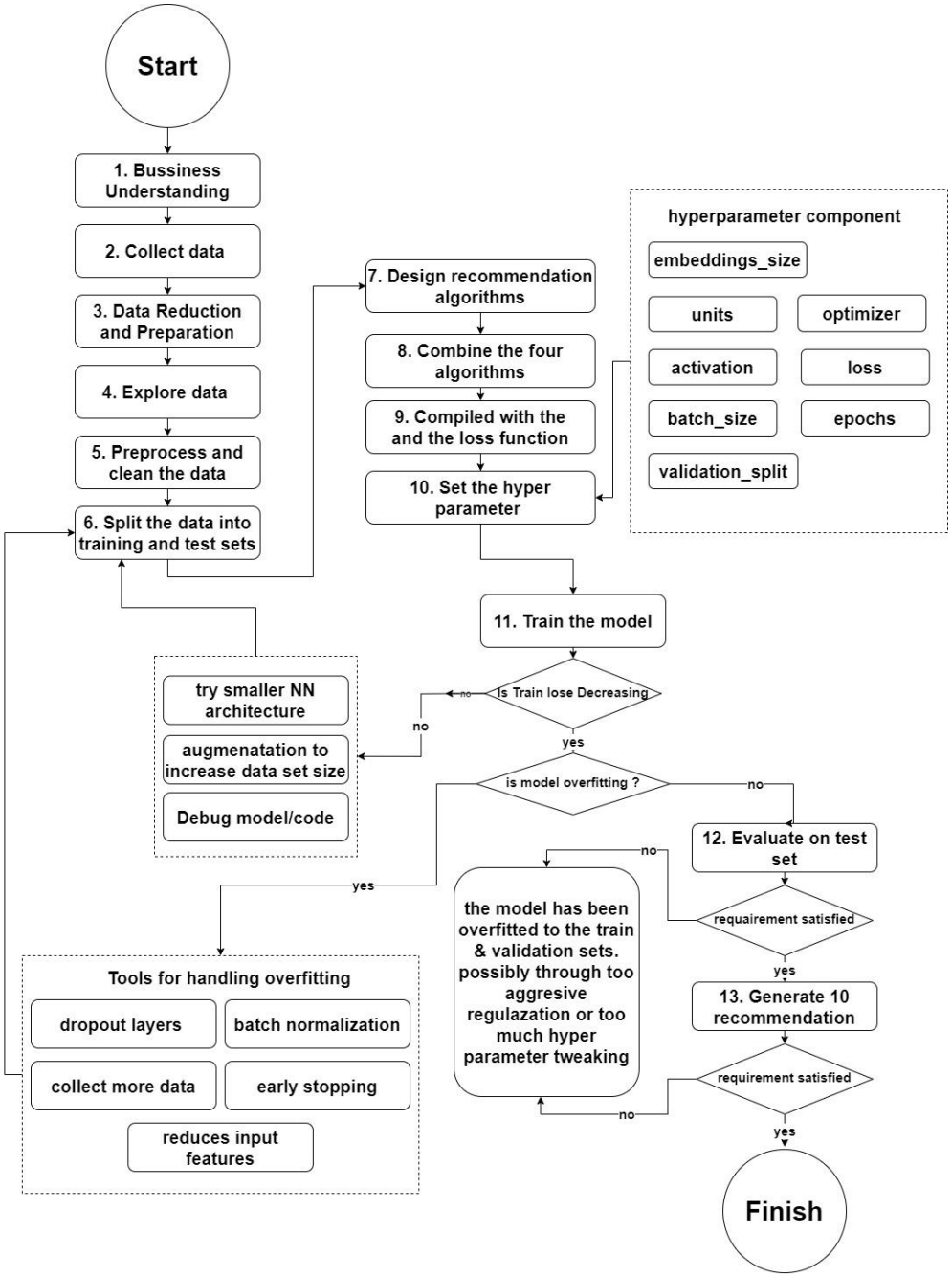


Figure 4 research flowchart

3.2Business understanding

Business understanding is a critical stage in the development of a health recommender system, encompassing a thorough understanding of the business context, objectives, and constraints by healthcare providers and stakeholders. This phase begins with stakeholder identification and progresses to defining business goals, understanding user needs, and identifying key performance indicators (KPIs) aligned with overarching objectives. It also involves assessing data availability, quality, and suitability, exploring domain-specific rules and constraints, and analyzing the competitive landscape to get insights and mitigate potential risks. By understanding business requirements upfront,

development of this system can make informed decisions throughout the process, ensuring that the result of hybrid recommender system effectively serves both business objectives and user needs.

To effectively understand the requirements for developing this health recommender system, we conducted a survey at Jogja International Hospital (JIH), focusing on the perspectives of the marketing team. The survey aimed to achieve the following objectives:

1. Understanding Goals: Identify the main objectives of the hospital marketing team in promoting healthcare services.
2. Assessing Challenges: Determine the challenges faced by the marketing team in their current decision-making and promotional efforts.
3. Exploring Decision-Making: Gain insight into how the marketing team makes decisions regarding service promotion and advertising.
4. Information Preferences: Understand the types of information and recommendations the marketing team finds most valuable in their promotional activities.
5. Checking System Requirements: Identify specific features or functionality desired in a recommendation system for healthcare promotion, such as data visualization tools or integration options.

3.3Data collection

This study obtained the clinical decision recommendation system data from the MIMIC-IV dataset on the physionet.org website(*MIMIC-IV v2.1*, n.d.). MIMIC-IV database contains real hospital stays for patients admitted to the Beth Israel Deaconess Medical Center (BIDMC) in Boston, MA, USA. It includes comprehensive information such as laboratory measurements, medications, vital signs, and more for each patient that we can access on the physionet.org website, as it is a public dataset. The dataset contains a large amount of data on hospital stays. It was chosen for this study because it provides a comprehensive and relevant dataset for developing a clinical decision recommendation system.

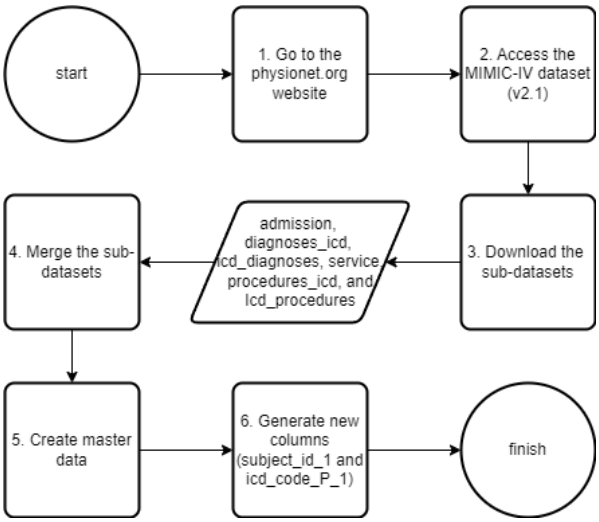


Figure 5 Data collection flowchart

We selected six relevant sub-datasets from MIMIC-IV to include in our master dataset: admission, diagnoses_icd, icd_diagnoses, service, procedures_icd, and Icd_procedures. The admission sub-dataset contains demographic and administrative information about each patient, including their subject_id, hadm_id, admission_type, admission_location, insurance, marital_status, ethnicity, gender, and anchor_age. The diagnoses_icd sub-dataset contains information on diagnoses made during the patient's hospital stay, including the seq_num, icd_code, and icd_version for each diagnosis. The icd_diagnoses sub-dataset contains the corresponding icd_title for each diagnosis, providing more detailed information. The service sub-dataset contains information on the type of service provided to the patient during their hospital stay, including the prev_service and curr_service. The procedures_icd sub-dataset contains information on procedures performed during the patient's hospital stay, including the seq_num_P, icd_code_P, and icd_version_P for each procedure. The Icd_procedures sub-dataset contains the corresponding long_title_P for each procedure, providing more detailed information.

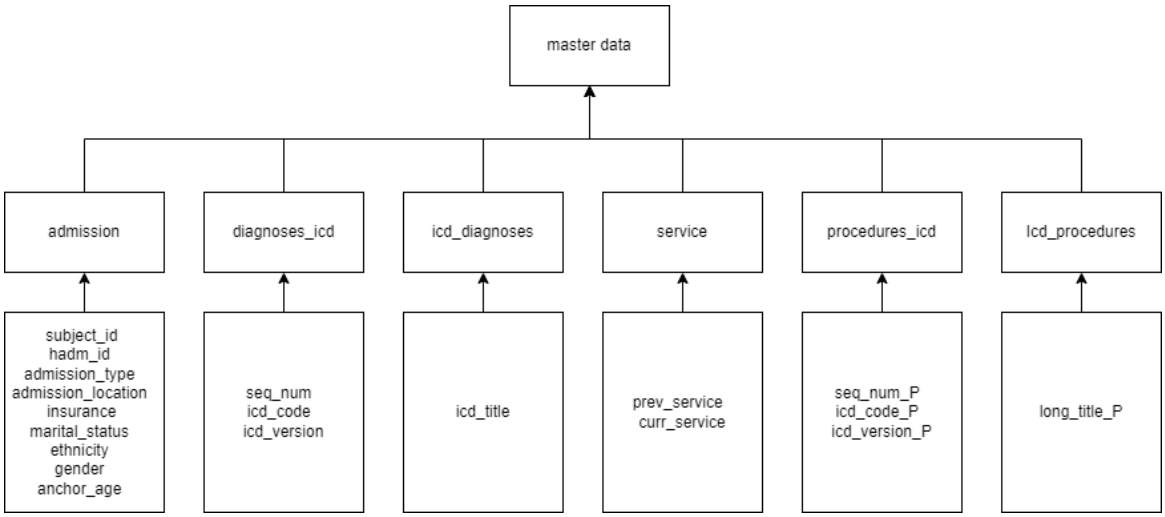


Figure 6 Data Collection

After gathering the data into one master data, we generated two new columns in the master data: subject_id_1 and icd_code_P_1. These two columns were created from the existing column, which is subject_id and icd_code_P using the pandas.cat.codes method, which helped to make the data more efficient regarding RAM usage in the following steps.

3.4Data Reduction and Preparation

After we gathered the master data, we reduce the data by reducing the rows in the procedure and service columns to a maximum of 5000 types of diagnoses, respectively. This was necessary due to hardware limitations, specifically the available RAM. By reducing the number of rows, we could avoid overwhelming the system with too much data, which would have hindered its performance.

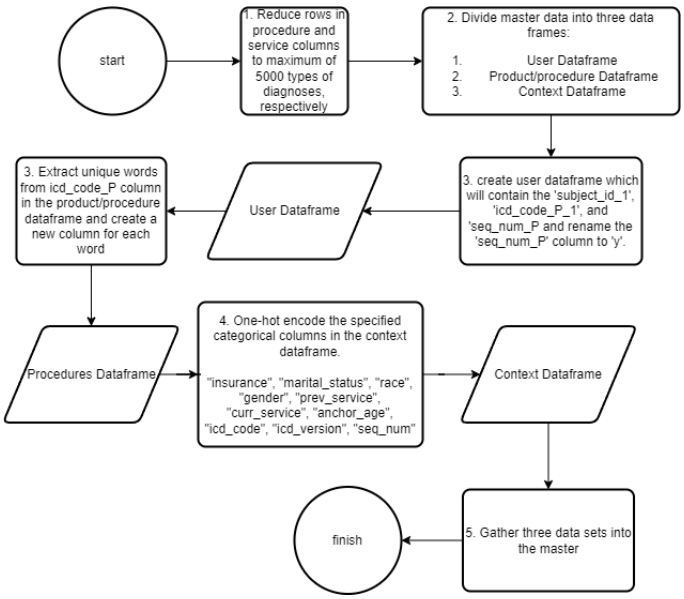


Figure 7 Dara reduction and preparation flowchart

After reducing the rows and organizing them, we divided the master data into three data frames:

1. User Dataframe,
2. Product/procedure Dataframe, and
3. Context Dataframe.

This was done better to facilitate the creation of the hybrid recommendation system because each of them needs a different input format due to the difference in their purpose. We gather three data sets into the master dataset again after we make the correct data format for each algorithm.

- The user data frame contains information about the clinical procedures history.
- The product/procedure data frame contains information about the recommended products or procedures, such as their characteristics and features.
- The context data frame contains information about the context in which the recommendations are being made, including any relevant factors or variables that may influence the recommendations such as age, insurance, marital status, gender, and ethnicity.

3.4.1 User DataFrame

The user data frame is created from the master data and contains only the columns subject_id_1, icd_code_P_1, and seq_num_P. In order to use a collaborative filtering recommendation algorithm, it is typically necessary to have our data in a matrix format (Aggarwal, 2016d), with rows representing individual patient-procedures interactions and columns representing the features of those interactions (subject_id_1, icd_code_P_1, seq_num_P). This format is often easier to work with when building a recommendation system as it allows the system to consider each individual rating separately, it would be organized as follows:

Tabel 1 Long Format

Subject_id	Procedures	Seq_number
1	procedures_1	3
1	procedures_2	4
1	procedures_3	5
2	procedures_1	2
2	procedures_2	4
2	procedures_3	5
3	procedures_1	1
3	procedures_2	3
3	procedures_3	2

3.4.2 Procedures DataFrame

. The product/procedure data frame contains the following columns from the master data: icd_code_P_1, icd_code_P, icd_version_P, and long_title_P. The product/procedure data frame contains ICD code information yhat represent the products or procedures being recommended.

To further preprocess this data, we extracted the unique words in the icd_code_P column for each row and created a new column for each word. We then filled each row in these new columns with a 0 or 1, indicating whether or not the word was present in the icd_code_P column for that row

For instance, the data looked like this:

Tabel 2 initial unextracted tabel

procedures	Icd_code
1	abc
2	def
3	ghi

After we extracted, the data would look like this:

Tabel 3 features extracted tabel

procedures	a	b	c	d	e	f	g	h	i
1	1	1	1	0	0	0	0	0	0
2	0	0	0	1	1	1	0	0	0
3	0	0	0	0	0	0	1	1	1

3.4.3 Context DataFrame

The context data frame is created from the master data and contains the following columns: insurance, marital_status , ethnicity, gender, anchor_age, seq_num, icd_code, icd_version, prev_service, curr_service. We use one-hot encode the specified categorical columns (insurance, marital_status, ethnicity, gender, icd_code, prev_service, and curr_service) and pass through the remainder of the columns. We apply the one-hot encoding to the categorical columns in the context data frame, which helps to prepare the data for use in the hybrid recommendation system.

One hot encoding is a method of encoding categorical variables as numerical data that can be used in machine learning models. It involves creating a new column for each unique category in a categorical feature and assigning a binary value (0 or 1) to indicate the presence or absence of that category in the original data(Hancock & Khoshgoftaar, 2020). One hot encoding is often used to preprocess categorical data for machine learning models because many algorithms cannot handle categorical data directly.

For instance, if the original data looked like this:

Tabel 4 initial categorical data

subject_id	diagnose
1	Flu
2	Headache
3	Diarrhea

After one hot encoding, the data would look like this:

Tabel 5 one-hot-encoded data

subject_id	diagnose_flu	diagnose_head ache	diagnose_diarrhea
1	1	0	0
2	0	1	0

3	0	0	1

3.5Data exploration

Data exploration is the process of analyzing and summarizing the main characteristics of a dataset, to gain a better understanding of the data and identify any patterns or trends that may be present. This involves several steps to explore and understand the features of a product or procedure in a our master data.

3.5.1 Data Distribution

We made plot figures to overview the distribution of the y variable, an assigned priority for procedures that occurred within the hospital stay and helped identify any potential outliers or patterns.

3.5.2 Correlation Between Features and Products

In this step, we generate the heatmap that will display the correlation coefficients between each feature of the product or procedure. The correlation coefficient measures the strength and direction of the linear relationship between each alphabet code in icd code. The values of the correlation coefficients will range from -1 to 1, where a value near 1 indicates a perfect positive correlation that will be presented as more dark color. By creating a heatmap, we can identify which features positively or negatively correlate with each product. This information can be helpful in the development of our understanding of which features are significant predictors of the product and can help in making decisions related to product development or marketing.

3.5.3 Ploting Procedures Features

This step involves creating a customized histogram plot to show the correlation between features (alphabet of the ICD code) in procedures for each patient. Each patient is plotted on the y-axis, and the product features (alphabets of the ICD code) are plotted on the x-axis. The plot shows the frequency of occurrence of each feature for each patient.

The plot is used to identify patterns and trends in the occurrence of specific features across different patients. It can help identify commonalities and differences in the procedures performed for different patients and can be used to analyze the effectiveness of different treatment approaches. Examining the plot can also identify any outliers or unusual occurrences in the data.

3.5.4 Plotting Sequence Nuumber and Anchor Ages

We generate Histogram Plot for Sequence Numbers and Anchor Ages involves creating a plot that displays the distribution of the number priority of a particular procedure that occurred during the hospital stay for each age group in the dataset. This visualization helps us to identify any patterns or trends in the data.

3.6 Data cleaning

outliers in the variable 'y' are removed by calculating the 1st and 99th percentile values of 'y'. Any value outside this range is considered an outlier and clipped to the closest percentile value. After removing outliers, the variable 'y' visualized through a histogram plot is created to visualize the distribution of 'y' after removing outliers. This step helped us eradicate any potential data issues, such as outliers or non-normal distribution, which may impact the subsequent analysis.

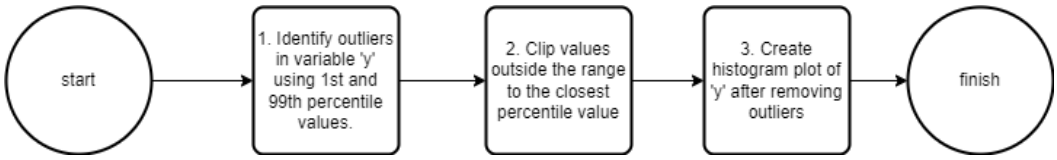


Figure 8 Data Cleaning Flowchart

3.7 Data standardization

It transformed the data into a mean of zero and a standard deviation of one. It is a common practice in data analysis and modeling, as it helps to compare variables on a standard scale and makes it easier to identify patterns and relationships between variables.

3.8Data splitting for training and testing

After we prepare data, we need to split the data into train and test. We do it so that we can then use it to train and test our recommendation system. Both datasets are being created from the user data frame. The train and test data frames are then merged with the procedures data frame and context data frame using the merge function.

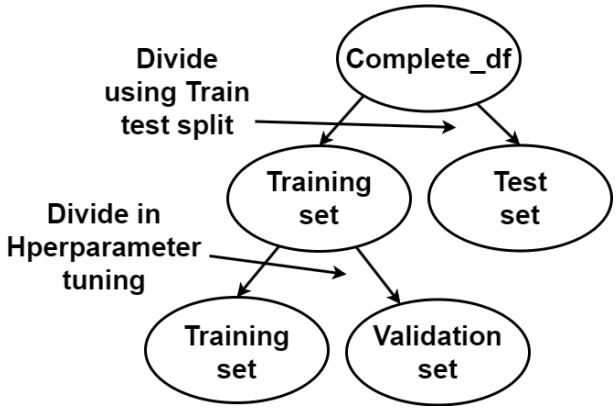


Figure 9 spliting data

To determine the split ratio, we will consider the number of null values in the dataframe and the impact that these null values may have on the model's performance. We may need to experiment with different split ratios to find the one that yields the best results.

3.9Design recommendation algorithms

Hybrid recommendation systems that utilize deep learning techniques can be very advanced and effective at making personalized recommendations. Deep learning techniques like neural networks can enable a recommendation system to learn and adapt to individual users' preferences and behavior and make highly accurate and personalized recommendations(Gündoğan & Kaya, 2022).

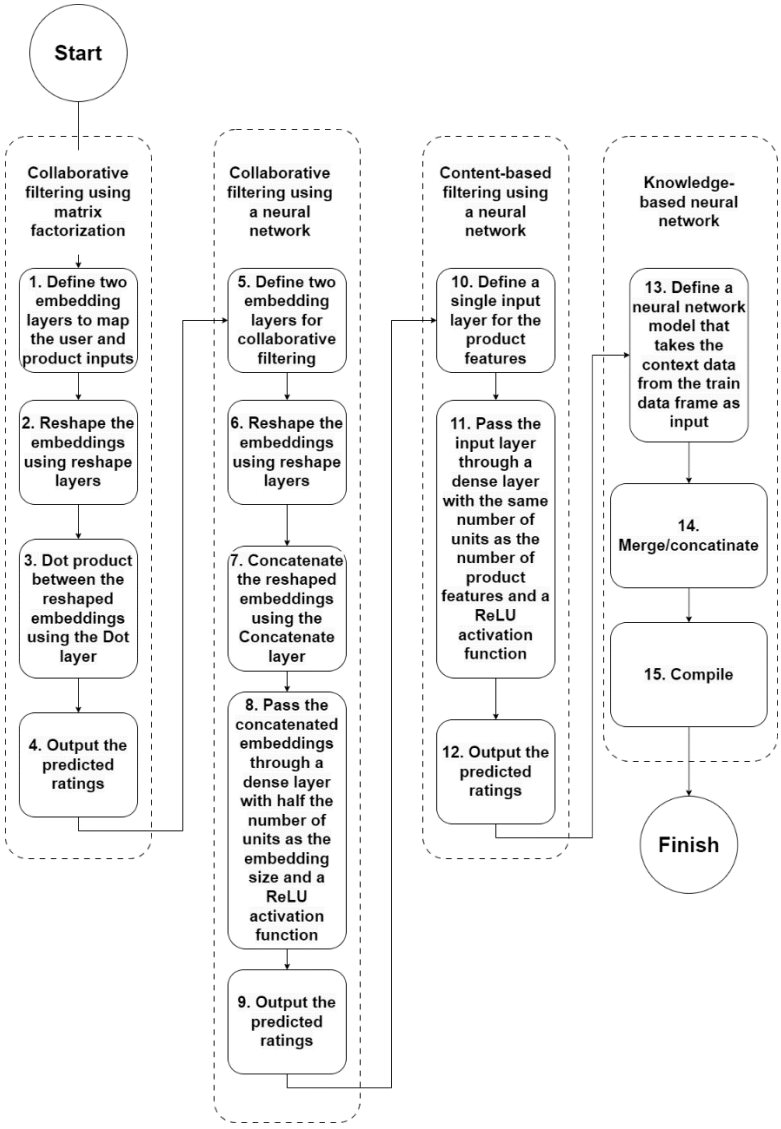


Figure 10 Design recommendation algorithms Flowchart

We will use neural networks to combine four recommendation algorithms in our hybrid recommendation system. These algorithms include collaborative filtering using matrix factorization, collaborative filtering using a neural network, content-based filtering using a neural network, and knowledge-based neural network. By using neural networks, we can learn and adapt to the preferences and needs of individual users and

make more accurate and diverse recommendations based on complex and diverse data. We are defining an artificial neural network using the functional API provided by Keras.

3.9.1 Collaborative filtering using matrix factorization.

Collaborative filtering using matrix factorization is a method of making recommendations based on the ratings or preferences of other users. It is commonly used in recommendation systems and has been successful in many applications(Bokde et al., 2015).

Matrix factorization represents the user-item interaction matrix as the product of two low-rank matrices: one for users and one for items. By doing this, the model learns the latent factors that explain the patterns in the rating matrix. The dot product between the low-rank user and item matrices is a similarity measure between a user and an item. This approach computes item-item similarities by comparing the embeddings of their corresponding columns in the item matrix and user-user similarities by comparing the embeddings of their corresponding rows in the user matrix.

For collaborative filtering using the matrix factorization algorithm, we define two embedding layers to map the user and product inputs to the embedding space. We are then reshaping the embeddings using reshape layers and taking the dot product between the reshaped embeddings using the Dot layer.

Learning the latent factors, collaborative filtering using matrix factorization is a simple linear operation that does not require an activation function to capture the desired information.

3.9.2 Collaborative filtering using a neural network

We improve the recommendation system by implementing Neural Collaborative Filtering. This model combines matrix factorization with the complexity of neural networks to make the most of the embedding space. It utilizes traditional collaborative filtering and incorporates a fully connected deep neural network to capture patterns and features that matrix factorization may miss. Training this model on the train data frame and using it to make recommendations to the user in the test data frame has proven that the show offers better performance(He et al., 2017).

Rather than computing the dot product, the collaborative filtering neural network concatenates the user and the item embeddings and feeds them into a dense layer. This layer learns non-linear interactions between the user and item features, and the resulting output can be used to predict the rating.

Using a neural network algorithm to map the user and product inputs to the embedding space, we define two embedding layers for collaborative filtering. We are then reshaping the embeddings using reshape layers and concatenating the reshaped embeddings using the Concatenate layer. We then pass the concatenated embeddings through a dense layer with half the number of units as the embedding size and a ReLU activation function.

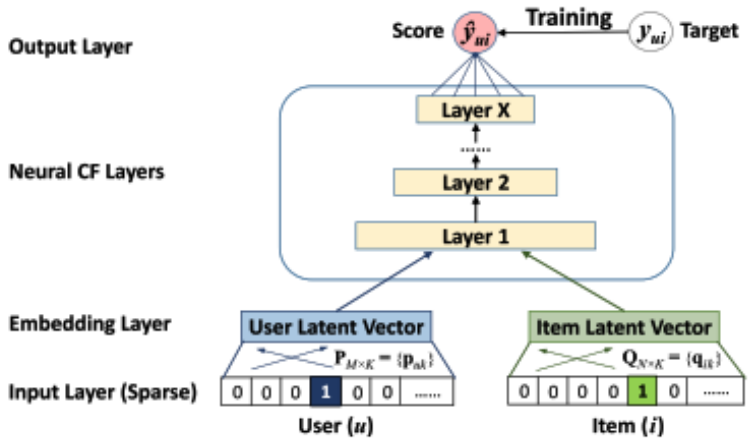


Figure 11 Neural Collaborative filtering framework(He et al., 2017)

For example, given input vectors for users and items, x_u and x_i, respectively, the output of the neural network can be calculated using the following equation:

$$y = f(W_1 * [x_u; x_i] + b_1) \quad (7)$$

Where :

- y is the output of the neural network,
- f is the activation function applied to the output of the first fully connected layer,
- W_1 and b_1 are the weights and biases of the first layer, respectively, and
- [x_u; x_i] represents the concatenation of the user and item vectors, x_u and x_i.

3.9.3 Content-based filtering using a neural network

We will create a content-based recommendation system using a neural network. To do this, we will create a neural network model that takes the characteristics and features of the items in the train data frame as input and predicts the ratings that users would give to these items. We will then train this model on the train data frame and use it to make recommendations.

Content-Based Filtering is used to find procedures that have similar characteristics to the ones a patient has received in the past. It learns the features

of the procedures and makes predictions based on the similarity between the features of the procedures.

For the content-based filtering using a neural network algorithm, we define a single input layer for the product features and pass it through a dense layer with the same number of units as the number of product features and a ReLU activation function.

For example, given a set of item features x_i, the output of the neural network can be calculated using the following equation:

$$y = f(W_1 * x_i + b_1) \quad (8)$$

Where :

- y is the output of the neural network
- f is the activation function of the first fully connected layer
- W_1 and b_1 are the weights and biases of the first fully connected layer
- x_i is the feature vector of the item being recommended

3.9.4 Knowledge-based neural network

We will incorporate knowledge-based recommendations into our hybrid system by creating a neural network model that takes the context data from the train data frame as input (e.g., insurance, marital status, ethnicity, gender) and uses this information to make user recommendations in the test data frame.

Knowledge-Based Filtering is used to find procedures recommended by medical professionals based on the context of the patient's medical history. It learns the features of the context and makes predictions based on the similarity between the features of the context.

For the knowledge-based neural network algorithm, we are defining a single input layer for the context features and passing it through a dense layer with the same number of units as the number of context features and a ReLU activation function.

For example, given a set of context features x_c, the output of the neural network can be calculated using the following equation:

$$y = f(W_1 * x_c + b_1) \quad (9)$$

Where :

- y is the output of the model
- f is the activation function of the first fully connected layer
- W_1 is the weight matrix of the first fully connected layer

- x_c is the input data, consisting of the context features (e.g., insurance, marital status, ethnicity, gender)
- b_1 is the bias vector of the first fully connected layer.

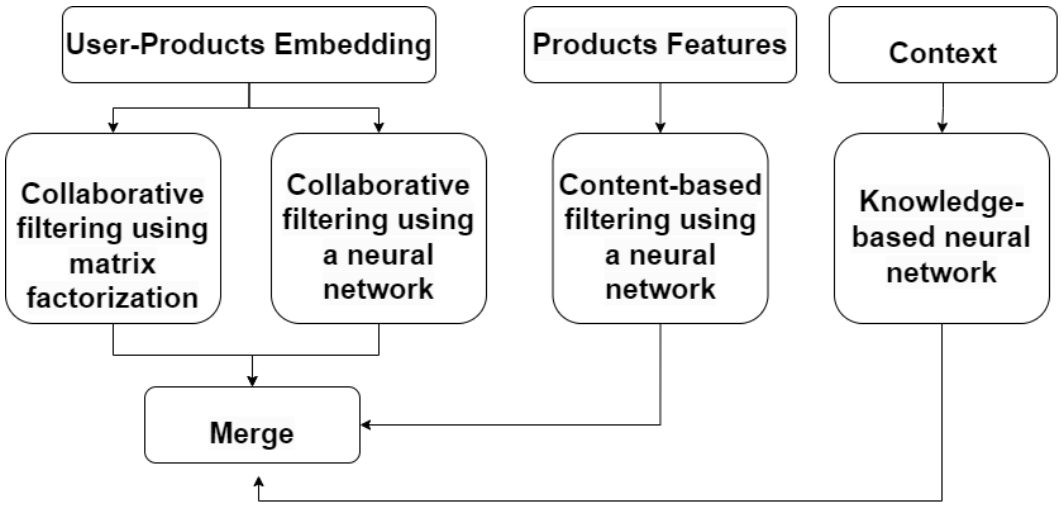
3.10 Combine the four algorithms

After we define the ulitimate four algorithm, the next step are using the Concatenate layer to merge the four algorithms' output and pass it through a final dense layer with a single unit and a linear activation function. This final output layer will provide the prediction for the recommendation system.

The four recommendation algorithms can be combined using a weighted sum of their respective prediction values. Let us say that the prediction values from the collaborative filtering using a matrix factorization algorithm are denoted as CF_MF, the prediction values from the collaborative filtering using a neural network are denoted as CF_NN, the prediction values from the content-based filtering using a neural network are denoted as CB_NN, and the prediction values from the knowledge-based filtering using a neural network is denoted as KB_NN. The final prediction values for the hybrid recommendation system can be calculated as follows:

$$\text{Prediction} = w1CF_MF + w2CF_NN + w3CB_NN + w4KB_NN \quad (10)$$

where w1, w2, w3, and w4 are weighting coefficients that can be determined through Adam optimization methods. The weighting coefficients represent the relative importance of each recommendation algorithm in the final prediction values. Here is what it will look like:



3.11 Compiled with optimizer and loss function

After creating the neural network architecture, the next step is to compile the model with an optimizer and a loss function. We will use the compile method on the model object to compile the model, which takes in several arguments. The first is to set the optimizer, the algorithm that will be used to adjust the weights and biases of the model to minimize the loss. We use the Adam optimizer.

The second is to set the loss function, which is the metric that will be used to measure how well the model is performing. In this case, we use the mean absolute error (MAE) loss function because MAE is easy to interpret and understand. It is the average error in the model's predictions(Willmott & Matsuura, 2005).

The third is to set metrics that will be used to evaluate the model's performance. In this case, we use the mean absolute percentage error (MAPE) metric, which measures the average absolute percentage difference between predicted and actual values. This makes it a useful tool for evaluating the performance of our model on both a practical and theoretical level (De Myttenaere et al., 2016). the compile method configures the model's learning process, including the optimizer, loss function, and any additional metrics we want to track.

3.12 Set the hyperparameter

We set the hyperparameters for our model by using the fit method on our train data. The input to this method includes the features and target columns from the train dataframe, as well as the number of epochs and batch size to use during training. We also specify that the data should be shuffled before each epoch and that the training progress should be displayed with a verbosity. Additionally, we set the validation split, which means the training data will be used for validation during training.

3.13 Train

The fit method trains the model on the training data and evaluates its performance on the validation data. This will train the model using the train data, with the features and target columns specified as input and output. The number of epochs, batch size, and other hyperparameters (such as shuffle and verbosity) will be set according to the values provided in the fit method. The model's progress will be displayed during training, and some parts of the training data will be used for validation.

Once the training is complete based on our judgment of the mean absolute percentage error (MAPE) value, which measures the average absolute percentage difference between the predicted and actual values, the model will be trained and ready for use in making recommendations.To track the training process, we set verbosity in hyperparameters. We can plot both the loss and MAPE into a line chart.

3.14 Evaluate

3.14.1 Training

We will use the training dataset that we split in earlier hyperparameter tuning as validate data in training phase. The record or history of training model are saved when it is complete. After the training phase done we can do investigation on our model and generate plot on it.

In order to validate the model, we can use the validation_split parameter in the fit function to split the training data into a training set and a validation set. The validation set is used to evaluate the model during training, and the model's performance on the validation set is used to tune the hyperparameters and decide when to stop training. By setting validation_split to 0.3, we are using 30% of the training data as the validation set and 70% as the training set. This allows us to monitor the model's performance on unseen data during training and make adjustments as needed.

3.14.2 Matrics evaluation

Test datasets that we split earlier are used at this stage. The model does not look this test data set, it mean that this data set we use as represent real world data and we will test our model using MAE and MAPE metrics on this datasets.

3.15 Generate 10 recommendation

In this project, we have implemented a recommendation system that combines collaborative filtering using matrix factorization, collaborative filtering using a neural network, content-based filtering using a neural network, and knowledge-based filtering using a neural network. The model has been trained and tested on a dataset of medical procedures and has shown good performance in terms of mean absolute error and mean absolute percentage error. While we successfully implemented the recommendation system, we could not implement it in a GUI or API due to limitations in our skills and resources. However, the system can be accessed through a Python prompt, such as Google Colab, to generate 10 recommendations to support clinical decision-making.

Chapter IV

Results and Discussion

4.1Business Understanding

We conducted a survey to understand the marketing team's goals, challenges, preferences, and needs at Jogja International Hospital the details of which are provided in the appendix.

a. Marketing Team Goals:
   The marketing team aims to enhance brand awareness, increase patient visits, and achieve specific objectives such as acquiring new patients and improving customer satisfaction.
b. Current Challenges:
   Challenges include competition in healthcare promotion and limited real-time campaign analysis. However, collaboration barriers with other hospital departments are minimal, and the team adapts through proactive monitoring.
c. Decision-Making Process:
   The team relies on data-driven insights, market research, and strategic planning for service promotion. Patient feedback is considered but not primary in decision-making.
d. Information Preferences:
   Valuable information includes service education by doctors, service videos, and patient testimonials. The team prioritizes evidence-based information and relies on customer relationship management (CRM) data.
e. System Requirements for Recommender System:
   The team desires personalized recommendation algorithms, customizable recommendations, and seamless integration with CRM systems. Addressing challenges in personalizing content for diverse patient demographics is crucial.

4.2Data collection

Our primary data source for this study was medical data from the MIMIC-IV v2.2 dataset, a publicly available medical database. We selected six relevant sub-datasets from MIMIC-IV to include in our master dataset: admission, diagnoses_icd, icd_diagnoses, service, procedures_icd, and Icd_procedures.

We imported libraries such as cudf, cuml, and cupy, commonly used in handling large-scale data in machine learning and data science. We read several CSV files from the MIMIC-IV v2.2 dataset, including admission information, patient information, diagnoses information, procedures information, and services information. These CSV files were stored in variables such as 'admissions', 'patients', 'diagnoses_icd'.

We merged some CSV files based on specific columns to create a new dataframe called 'master_data'. We used the 'matplotlib' library to plot three bar graphs that show the frequency of icd_code_P procedures performed on patients. The first bar graph shows the frequency of all icd_code_P procedures, while the second and third bar graphs show the frequency of the top 300 and top 30 icd_code_P procedures, respectively. The code also printed the shape of the 'master_data' dataframe and the number of missing values in each column of the 'master_data' dataframe.

	subject_id	hadm_id	seq_num	icd_code	icd_versic	long_title	admission	admission	insurance	marital_st	...	anchor_ag	prev_serv	curr_servi	seq_num_	icd_code_	icd_versic	long_title	icd_code_	icd_code_	subject_id
0	18670506	29377271	7	V160	9	Family his	AMBULATI	PROCEDU	Other	MARRIED	...	64	0	OMED	1	10	9	Implantat	20730	0	58534
1	11441519	27173729	4	78701	9	Nausea w	AMBULATI	PROCEDU	Other	MARRIED	...	57	0	OMED	1	10	9	Implantat	5525	0	9682
2	11441519	21479013	3	5715	9	Cirrhosis c	AMBULATI	PROCEDU	Medicaid	MARRIED	...	57	0	OMED	1	10	9	Implantat	3541	0	9682
3	11419849	22603594	8	5715	9	Cirrhosis c	AMBULATI	PROCEDU	Other	SINGLE	...	57	0	OMED	1	10	9	Implantat	3541	0	9521
4	18670506	28816971	1	1550	9	Malignant	AMBULATI	PROCEDU	Other	MARRIED	...	64	0	OMED	1	10	9	Implantat	457	0	58534
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
402768	16440799	28713670	6	N400	10	Benign pr	ELECTIVE	PHYSICIAN	Medicare	MARRIED	...	82	0	OMED	1	XW0DXR5	10	Introducti	15022	9698	43456
402769	13274858	28451991	4	I69354	10	Hemipleg	URGENT	PROCEDU	Medicare	0	...	50	0	V5URG	5	XY0VX83	10	Extracorp	11573	9699	22105
402770	13274858	28451991	16	Z931	10	Gastrosto	URGENT	PROCEDU	Medicare	0	...	50	0	V5URG	5	XY0VX83	10	Extracorp	22543	9699	22105
402771	13274858	28451991	6	L7632	10	Postproce	URGENT	PROCEDU	Medicare	0	...	50	0	V5URG	5	XY0VX83	10	Extracorp	13116	9699	22105
402772	13274858	28451991	12	D631	10	Anemia in	URGENT	PROCEDU	Medicare	0	...	50	0	V5URG	5	XY0VX83	10	Extracorp	8511	9699	22105

402773 rows × 22 columns

Figure 12 Master Data

Table 2 The shape of the 'master_data' dataframe

	Row	Column	Description
master_data	1.2741.604	19	Shape
icd_code	22637		Value counts
icd_code_P	12574		Value counts

We also created a histogram using matplotlib to display the distribution of the procedures and diagnose column in the master dataset. The x-axis shows the range of values procedures and diagnoses of the column, and the y-axis shows the frequency of those values.

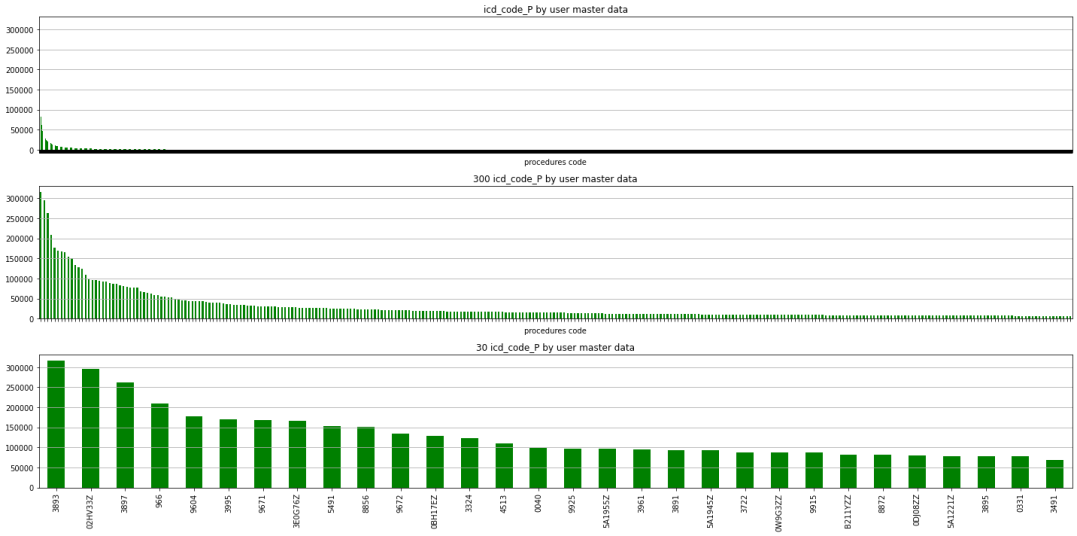


Figure 13 procedures master data

```
import cudf, cuml, cupy

admissions = cudf.read_csv('/content/drive/MyDrive/MIMIC-IV
v2.2/admissions.csv', usecols=['subject_id', 'hadm_id',
'admission_type', 'admission_location', 'insurance',
'marital_status', 'race'])
patients = cudf.read_csv('/content/drive/MyDrive/MIMIC-IV
v2.2/patients.csv', usecols=['subject_id','gender',
'anchor_age'])
diagnoses_icd = cudf.read_csv('/content/drive/MyDrive/MIMIC-IV
v2.2/diagnoses_icd.csv', usecols=['subject_id',
'hadm_id','icd_code', 'icd_version', 'seq_num'])
d_icd_diagnoses = cudf.read_csv('/content/drive/MyDrive/MIMIC-IV
v2.2/d_icd_diagnoses.csv',
usecols=['icd_code','icd_version','long_title'])
services = cudf.read_csv('/content/drive/MyDrive/MIMIC-IV
v2.2/services.csv', usecols=['subject_id',
'hadm_id','prev_service', 'curr_service'])
procedures_icd = cudf.read_csv('/content/drive/MyDrive/MIMIC-IV
v2.2/procedures_icd.csv', usecols=['subject_id',
'hadm_id','icd_code', 'icd_version', 'seq_num'])
d_icd_procedures = cudf.read_csv('/content/drive/MyDrive/MIMIC-IV
v2.2/d_icd_procedures.csv',
usecols=['icd_code','icd_version','long_title'])

demografi = admissions.merge(patients, on='subject_id')
diagnoses = diagnoses_icd.merge(d_icd_diagnoses,
on=['icd_code','icd_version'])
```

```
procedures = procedures_icd.merge(d_icd_procedures,
on=['icd_code','icd_version'])
procedures = procedures.rename(columns={'icd_code': 'icd_code_P',
'icd_version': 'icd_version_P', 'seq_num':
'seq_num_P','long_title': 'long_title_P'})
diagnose = diagnoses.merge(demografi, on=['subject_id',
'hadm_id'])
diagnose = diagnose.merge(services, on=['subject_id', 'hadm_id'])
diagnose = diagnose.fillna(0)
diagnose.set_index(['subject_id','hadm_id'], inplace=True)
procedures.set_index(['subject_id','hadm_id'], inplace=True)
master_data = diagnose.join(procedures,
how='outer').reset_index()
master_data = master_data.dropna()
#print(procedures.isna().sum())
#print(diagnose.isna().sum())
print(master_data.isna().sum())
print(master_data.shape)
```

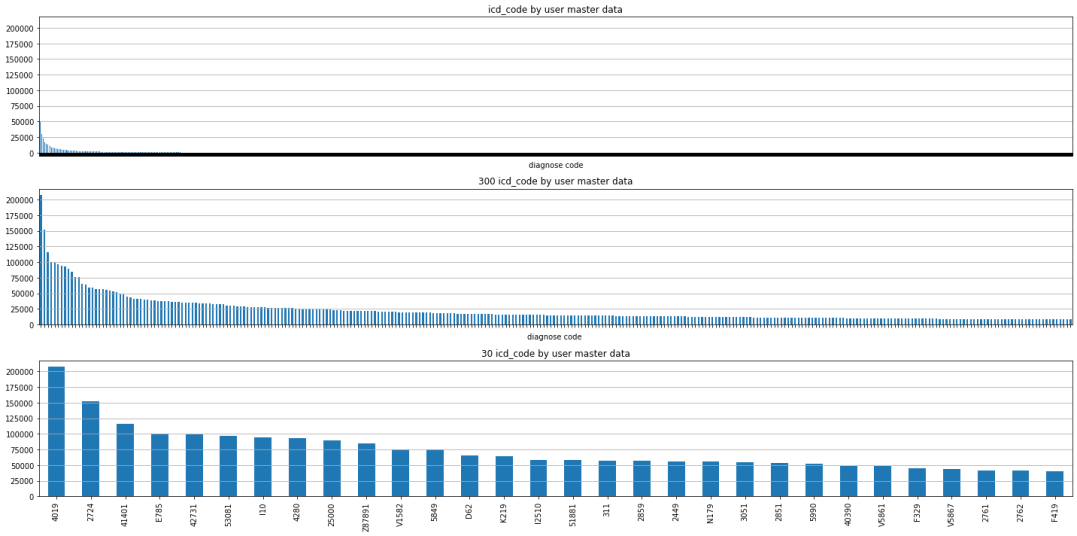


Figure 14 diagnose master data

We also created a histogram using matplotlib to display the distribution of the "curr_service" column in the master dataset. The x-axis shows the range of values of the "curr_service" column, and the y-axis shows the frequency of those values.

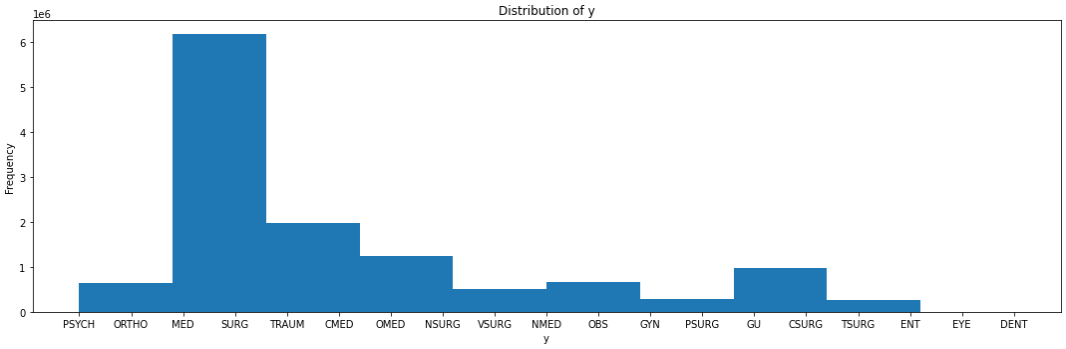


Figure 15 Y or seq_num distribution Master Data

4.3Data reduction and preparation

We are preparing the data for a hybrid recommendation system. We have problem that the master data was too big, and our machine could not handle it. We decided to reduce the data to overcome this problem. we reduce the data by reducing the number of rows in the procedure and service columns to a maximum of 5000 types of diagnoses, respectively. To do reduction we defined the number of bins as n_bins = 5000. Then, we selected the n_bins most frequent unique numerical and stored the index of those values in the top_n_codes variable. The master DataFrame was filtered only to include rows where the 'icd_code_num' column is one of the selected values. This helped to reduce the dimensionality of the dataset by focusing only on the most frequent icd_codes.

we are reducing the size of the master_data DataFrame by grouping the data by curr_service, icd_code, and icd_code_P columns and randomly sampling a maximum of n_max rows from each group. We use the apply method to apply the random sampling to each group, and we store the resulting sampled data in a new DataFrame called master_data.

```
# Create a new column 'icd_code_num' that assigns a numerical
value to each unique 'icd_code' value
master_data['icd_code_num'] =
master_data['icd_code'].astype('category').cat.codes

# Count the number of occurrences of each unique numerical value
in the 'icd_code_num' column
value_counts = master_data['icd_code_num'].value_counts()

# Define the number of bins
n_bins = 5000

# Select the `n_bins` most frequent unique numerical values
top_n_codes = value_counts.nlargest(n_bins).index
```

```
# Filter the original DataFrame to only include rows where the
'icd_code_num' column is one of the selected values
master_data =
master_data[master_data['icd_code_num'].isin(top_n_codes)]

# Group data by current service and randomly sample a certain
number of rows
n_max = 40000
master_data = master_data.groupby('curr_service').apply(lambda x:
x.sample(n=min(n_max, len(x))))

# Reset the index and drop unnecessary columns
master_data  = master_data.reset_index(level=[0,1], drop=True)

# Group data by ICD code and randomly sample a certain number of
rows
n_max = 1000
master_data = master_data.groupby('icd_code').apply(lambda x:
x.sample(n=min(n_max, len(x))))

# Reset the index and drop unnecessary columns
master_data = master_data.reset_index(level=[0,1], drop=True)

# Group data by current service and randomly sample a certain
number of rows
n_max = 1000
master_data = master_data.groupby('icd_code_P').apply(lambda x:
x.sample(n=min(n_max, len(x))))

# Reset the index and drop unnecessary columns
sampled_data = master_data.reset_index(level=[0,1], drop=True)
```

Table 3 Reduced master data shape

Metric	Value
Number of Rows	402,773
Number of Unique icd_codes	4,995
Number of Unique icd_code_P	9,7
Number of Unique curr_service	19
Total number of curr_service	402,773

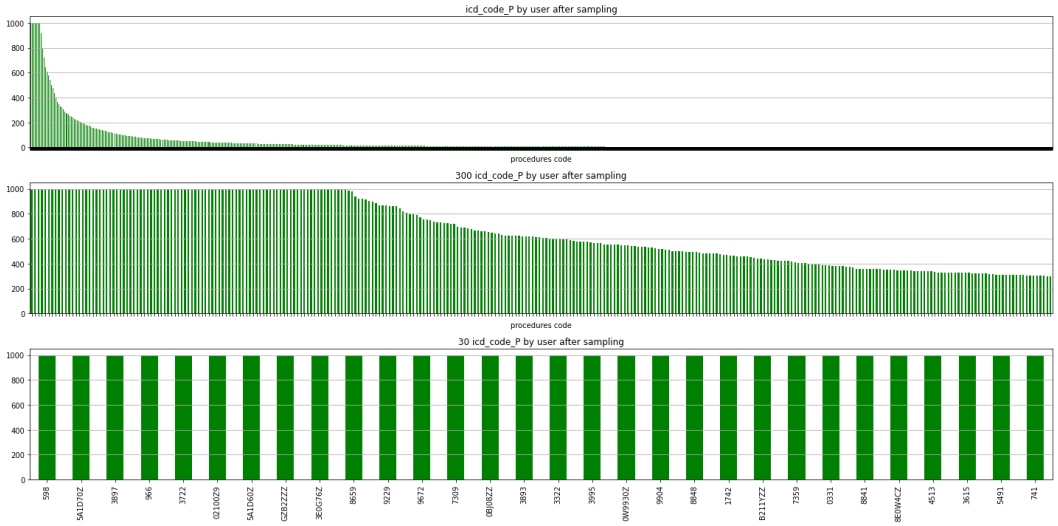


Figure 16 Procedures after reduction

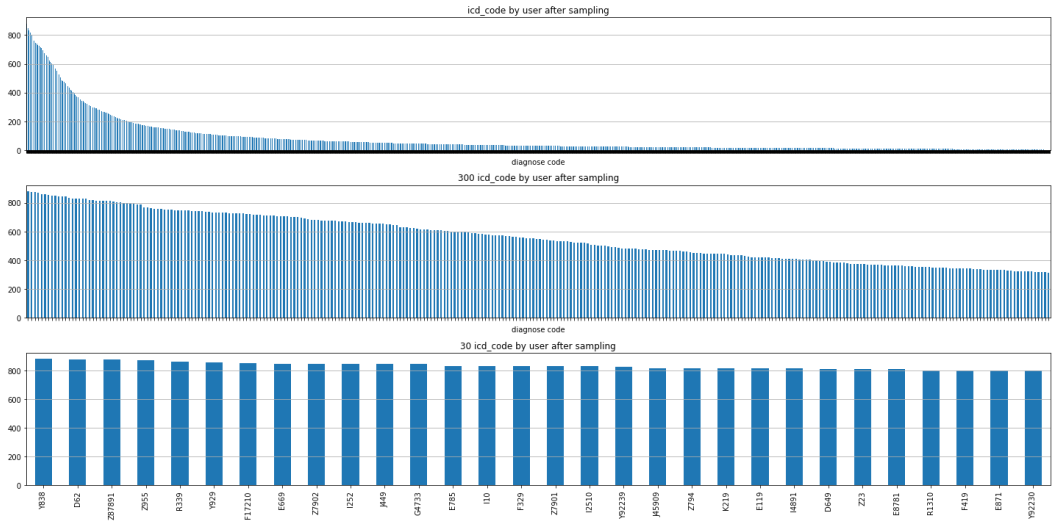


Figure 17 Diagnose after reduction

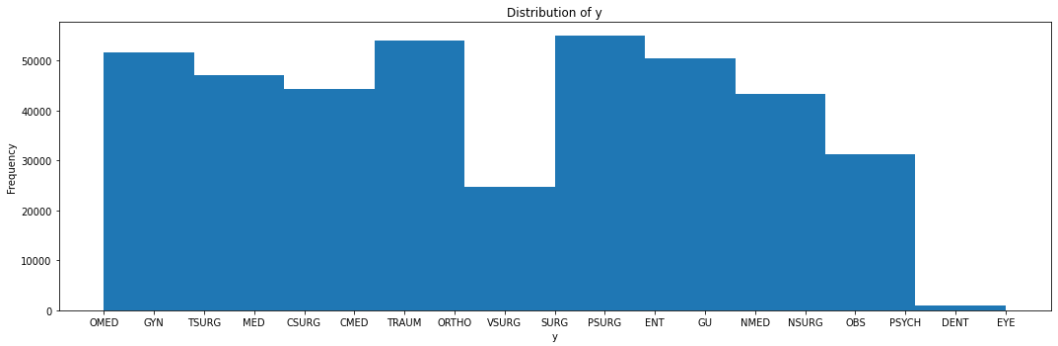


Figure 18 y distribution after reduction

```
sampled_data['icd_code_P_1']=sampled_data['icd_code_P']
sampled_data['subject_id_1']=sampled_data['subject_id']

#Convert icd_code_P_1 and subject_id_1 to category
sampled_data['icd_code_P_1'] =
sampled_data['icd_code_P_1'].astype('category')
sampled_data['subject_id_1'] =
sampled_data['subject_id_1'].astype('category')

#Assign new codes for icd_code_P_1 and subject_id_1
sampled_data['icd_code_P_1'] = sampled_data['icd_code_P_1'].cat.codes
sampled_data['subject_id_1'] = sampled_data['subject_id_1'].cat.codes
```

also we performs some basic data cleaning tasks such as converting categorical columns to numerical codes, dropping unnecessary columns, and plotting the frequency distributions. To prepare our data for recommendation algorithms, we then performed some data preprocessing by divide the master data into three data frames:

1. the user data frame,
2. the product/procedure data frame, and
3. the context data frame.

After we prepare the data by manipulate manipulate using varoius technique we concatenates the three data frames and creates a complete data frame for the recommendation system. The resulting complete data frame contains information about the system users, including their demographics and clinical procedures history (user data frame), information about the recommended products or procedures, such as their characteristics and features (procedures data frame), and information about the context in which the recommendations are being made (context data frame). This data frame can be used as input to a hybrid recommendation system to generate recommendations for clinical procedures based on user demographics and clinical history.

	subject_id_1	icd_code_P_1	y	icd_versic	H		8	X		3	S	D	...	icd_code_	icd_code_	icd_code_	icd_code_	icd_code_	icd_code_	icd_code_	icd_code_	icd_code_	icd_code_
0	58534	0	1	9	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	9682	0	1	9	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	9682	0	1	9	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	9521	0	1	9	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	58534	0	1	9	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
402768	43456	9698	1	10	0	0	2	0	0	1	...	0	0	0	0	0	0	0	0	0	0	0	0
402769	22105	9699	5	10	0	1	2	1	0	0	...	0	0	0	0	0	0	0	0	0	0	0	0
402770	22105	9699	5	10	0	1	2	1	0	0	...	0	0	0	0	0	0	0	0	0	0	0	0
402771	22105	9699	5	10	0	1	2	1	0	0	...	0	0	0	0	0	0	0	0	0	0	0	0
402772	22105	9699	5	10	0	1	2	1	0	0	...	0	0	0	0	0	0	0	0	0	0	0	0

402773 rows × 5117 columns

Figure 19 Complete data after preparation

4.2.1 User Dataframe

The user data frame contains information about the clinical procedures history. We created a user data frame from the master data, which contains only the columns subject_id_1, icd_code_P_1, and seq_num_P. Since collaborative filtering recommendation algorithms require data in a matrix format, with rows representing individual patient-procedures interactions and columns representing the features of those interactions, we needed to pivot our data to transform it into the appropriate format. We used the pivot_table function to reshape the data in the user data frame so that the subject_id_1 values become the index, the icd_code_P_1 values become the column labels, and the seq_num_P values become the values in the resulting pivot table.

	subject_ic	icd_code_	y
0	58534	0	1
1	9682	0	1
2	9682	0	1
3	9521	0	1
4	58534	0	1
…	…	…	…
402768	43456	9698	1
402769	22105	9699	5
402770	22105	9699	5
402771	22105	9699	5
402772	22105	9699	5
402773 rows × 3 columns			

Figure 20 User dataframe

4.2.2 Product/procedure Dataframe

The product/procedure data frame contains information about the recommended products or procedures, such as their characteristics and features. We converted the icd_code_P column to a string type and obtained a list of unique characters present in the column. We then created new columns for each unique character and updated them with the count of occurrences for each character. Finally, we dropped the original icd_code_P column from the dataframe.

	icd_code_	icd_versic	long_title_P	Z	W	F	R	S	T	C	…	P	V	7	1	5	X	G	2	U	H
0	0	9	Implantation of chemotherapeutic agent	0	0	0	0	0	0	0	…	0	0	0	1	0	0	0	0	0	0
1	0	9	Implantation of chemotherapeutic agent	0	0	0	0	0	0	0	…	0	0	0	1	0	0	0	0	0	0
2	0	9	Implantation of chemotherapeutic agent	0	0	0	0	0	0	0	…	0	0	0	1	0	0	0	0	0	0
3	0	9	Implantation of chemotherapeutic agent	0	0	0	0	0	0	0	…	0	0	0	1	0	0	0	0	0	0
4	0	9	Implantation of chemotherapeutic agent	0	0	0	0	0	0	0	…	0	0	0	1	0	0	0	0	0	0
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
402768	9698	10	Introduction of Venetoclax Antineoplastic into…	0	1	0	1	0	0	0	…	0	0	0	0	1	2	0	0	0	0
402769	9699	10	Extracorporeal Introduction of Endothelial Dam…	0	0	0	0	0	0	0	…	0	1	0	0	0	2	0	0	0	0
402770	9699	10	Extracorporeal Introduction of Endothelial Dam…	0	0	0	0	0	0	0	…	0	1	0	0	0	2	0	0	0	0
402771	9699	10	Extracorporeal Introduction of Endothelial Dam…	0	0	0	0	0	0	0	…	0	1	0	0	0	2	0	0	0	0
402772	9699	10	Extracorporeal Introduction of Endothelial Dam…	0	0	0	0	0	0	0	…	0	1	0	0	0	2	0	0	0	0
402773 rows × 37 columns																					

Figure 22 Procedures dataframe

```
Index(['icd_version_P', 'Z', 'W', 'F', 'R', 'S', 'T', 'C', '6', '0', 'Y', 'J', 'L', 'Q', '3', 'E', 'A', 'K', 'D', '8', 'M', 'B', '4', '9', 'N', 'P', 'V', '7', '1', '5', 'X', 'G', '2', 'U', 'H'], dtype='object')
```

Figure 21 Procedures features

4.2.3 Context Dataframe

The context data frame contains information about the context in which the recommendations are being made, including any relevant factors or variables that may influence the recommendations such as age, insurance, marital status, gender, and ethnicity. One-hot encoding is a method of encoding categorical variables as numerical data that can be used in machine learning models. This helps prepare the data for use in the hybrid recommendation system, which aims to provide personalized clinical recommendations for healthcare professionals, hospitals, and patients.

```
Index( ['anchor_age', 'icd_version', 'seq_num', 'insurance_Medicaid',
       'insurance_Medicare', 'insurance_Other', 'marital_status_0',
       'marital_status_DIVORCED', 'marital_status_MARRIED', 'marital_status_SINGLE', ...
       'icd_code_Z982', 'icd_code_Z9861', 'icd_code_Z9862', 'icd_code_Z9884',
       'icd_code_Z9889', 'icd_code_Z98890', 'icd_code_Z9911', 'icd_code_Z992',
       'icd_code_Z993', 'icd_code_Z9981'], dtype='object', length=5079)
```

Figure 24 Context Features

	anchor_ag	icd_versic	seq_num	insurance	insurance	insurance	marital_st	marital_st	marital_st	marital_st	...	icd_code_	icd_code_	icd_code_	icd_code_	icd_code_	icd_code_	icd_code_	icd_code_	icd_code_	icd_code_
0	64	9	7	0	0	1	0	0	1	0	...	0	0	0	0	0	0	0	0	0	0
1	57	9	4	0	0	1	0	0	1	0	...	0	0	0	0	0	0	0	0	0	0
2	57	9	3	1	0	0	0	0	1	0	...	0	0	0	0	0	0	0	0	0	0
3	57	9	8	0	0	1	0	0	0	1	...	0	0	0	0	0	0	0	0	0	0
4	64	9	1	0	0	1	0	0	1	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
402768	82	10	6	0	1	0	0	0	1	0	...	0	0	0	0	0	0	0	0	0	0
402769	50	10	4	0	1	0	1	0	0	0	...	0	0	0	0	0	0	0	0	0	0
402770	50	10	16	0	1	0	1	0	0	0	...	0	0	0	0	0	0	0	0	0	0
402771	50	10	6	0	1	0	1	0	0	0	...	0	0	0	0	0	0	0	0	0	0
402772	50	10	12	0	1	0	1	0	0	0	...	0	0	0	0	0	0	0	0	0	0

402773 rows × 5079 columns

Figure 23 Context dataframe

4.3 Data exploration

4.3.1 Data Distribution

We generate a histogram of the 'y' column in the 'complete_df' DataFrame, corresponding to the 'seq_num_P' column from the original data. The histogram shows the frequency distribution of the 'y' values, which are counts of medical procedures performed for each patient. We use a histogram to explore the distribution of the target variable 'y' in the complete_df DataFrame. The histogram shows the frequency of values in the 'y' column.

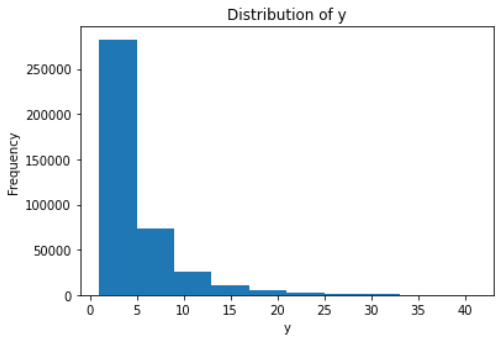


Figure 25 y distribution toward seq_num

We look at the histogram that the distribution is not good. We need to reduce the disparity of this data and decide to handle this outlier in the next step.

4.3.2 Correlation Between Features and Products

We check the correlation between the different procedures and features. It seems that some clusters of procedures are highly correlated with each other and some are not significantly correlated with any other procedure. This could be useful for feature selection, as we may be able to remove some of the less critical procedures without losing too much information.

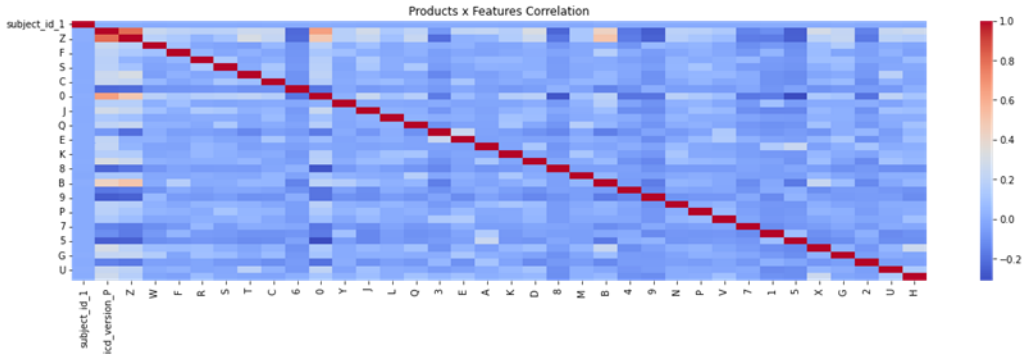


Figure 26 Procedures towards Featues correlation

4.3.3 Plotting Procedures Features

We check which procedures are present or absent for each subject. As expected, many procedures are only performed on a small number of subjects, and many are not performed at all.

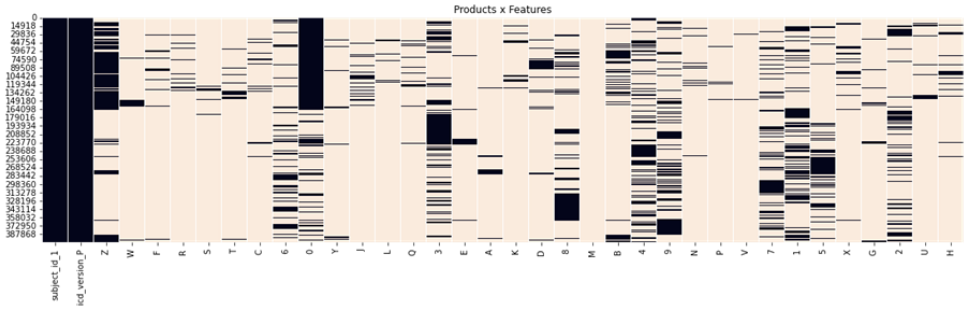


Figure 27 Features distribution in each patient

We make the code create a grid of subplots and iterate through each feature column in the procedures1 dataframe, plotting a histogram of each feature and setting the subplot title, xlabel, and ylabel accordingly. The resulting plot shows the distribution of each feature in procedures1.

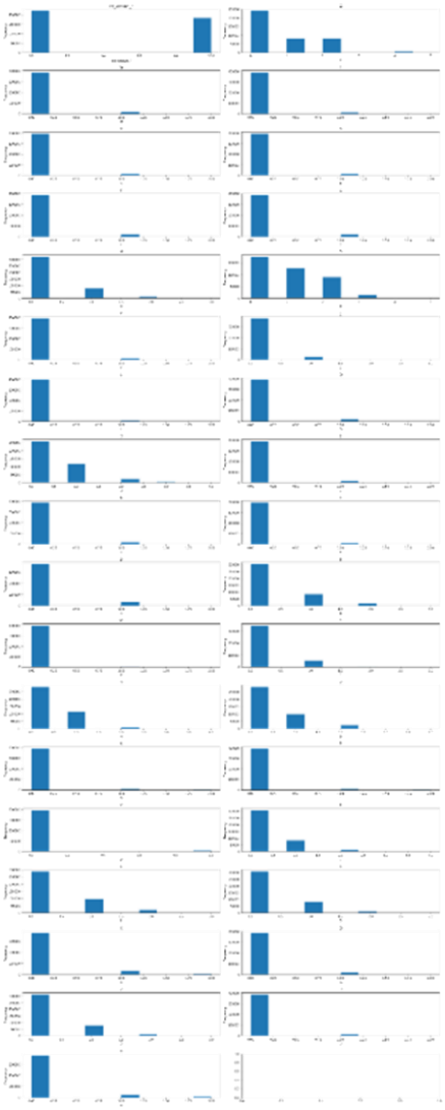


Figure 28 Features Distribution frequency

.

.

4.3.4 Plotting Sequence Number and Anchor Ages

We make plot that represents the distribution of anchor_age for a unique value of seq_num in the context dataframe. The sequence numbers are obtained by calling the unique() method on the seq_num column of the context dataframe. Then, in the for loop, the code filters the context dataframe to only include rows with the current sequence number and plots a histogram of the anchor_age feature for that sequence number in the current subplot. The subplot title is set to indicate the current sequence number.

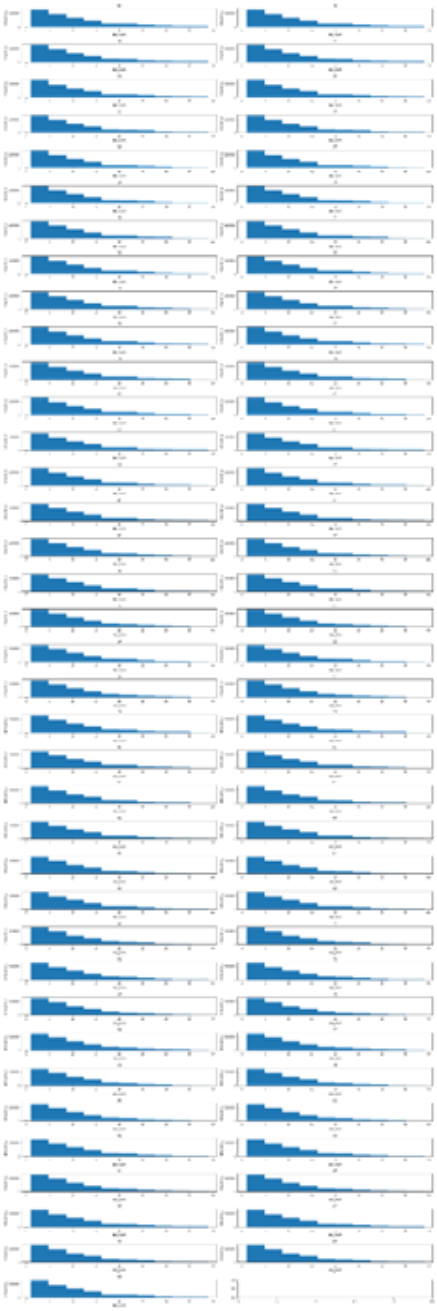


Figure 29 the distribution of anchor_age for a unique value of seq_num

4.4Data cleaning

We realize that we have outliers in the variable 'y'. To remove outliers in our data set, we set the values in the 'y' column that are less than the 1st percentile to the 1st percentile and values greater than the 99th percentile to the 99th percentile both are removed. This will limit the range of values in the 'y' column and remove extreme values that could skew the analysis.

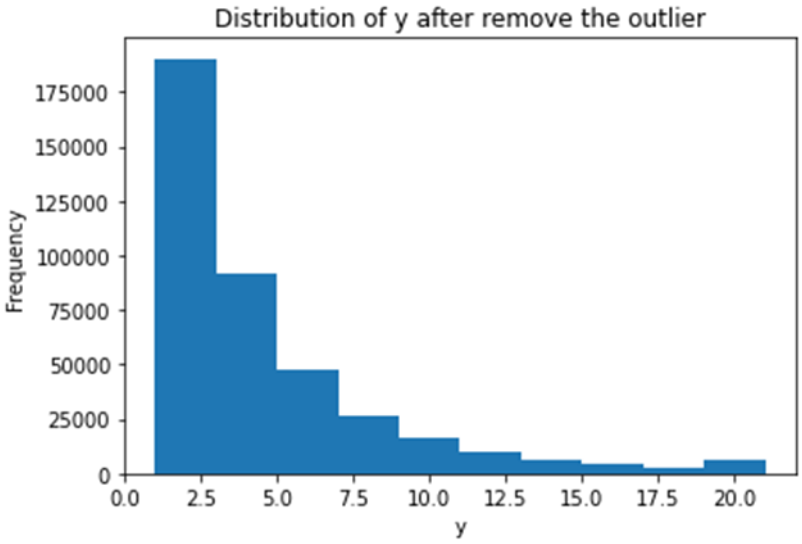


Figure 30 y distribution after data cleaning.

```
# remove outlier
q1 = complete_df['y'].quantile(0.01)
q3 = complete_df['y'].quantile(0.99)
complete_df['y'] = complete_df['y'].clip(lower=q1, upper=q3)
```

4.5Data Standardization

Standardize the numerical columns of the complete_df dataframe uisng sci-kit-learn library. Standardization is a common data preprocessing step that involves transforming the data with a mean of 0 and a standard deviation of 1.

First creates a list of columns to standardize, then use StandardScaler object from sci-kit-learn and fits it to the data using the fit_transform method, which returns a standardized version. This standarized version was the final dataframe that can be used for machine learning modeling.

```
from sklearn.preprocessing import StandardScaler

# Create a list of columns to standardize
numerical_columns = complete_df.select_dtypes(include=['float32',
'int32']).columns
columns_to_standardize = [col for col in numerical_columns if col
not in ['subject_id_1', 'icd_code_P_1', 'icd_version_P']]

# Initialize the StandardScaler object
scaler = StandardScaler()

# Fit and transform the data
```

```
complete_df[columns_to_standardize] =
scaler.fit_transform(complete_df[columns_to_standardize].to_panda
s())
```

4.6Design Artificial Neural Network

The developed ANN architecture for connecting four filtering algorithms consists of several components that are connected from one layer to another. These components are Collaborative Filtering, Matrix Factorization, Neural Network, Content-Based Filtering, Knowledge-Based Filtering, and Output layer. The model is designed with the following specifications: :

1. Collaborative Filtering:
    - Input layer: two separate inputs for users and products represented by their IDs.
    - Matrix Factorization:
        - Embedding layer for users with the specified number of dimensions (embeddings_size).
        - Embedding layer for products with the specified number of dimensions (embeddings_size).
        - Reshape layer to flatten the embeddings.
        - Dot product layer to combine the embeddings and compute the similarity between users and products.
    - Neural Network:
        - Embedding layer for users with the specified number of dimensions (embeddings_size).
        - Embedding layer for products with the specified number of dimensions (embeddings_size).
        - Reshape layer to flatten the embeddings.
        - Concatenate layer to merge the user and product embeddings.
        - Dense layer with ReLU activation function to perform a nonlinear transformation of the concatenated embeddings.
2. Content-Based Filtering:
    - Input layer: a single input for the product features.
    - Dense layer with ReLU activation function to transform the input features into a dense representation.

3. Knowledge-Based Filtering:
    - Input layer: a single input for the contextual information.
    - Dense layer with ReLU activation function to transform the input context into a dense representation.
4. Output:
    - Concatenate layer to merge the output from the collaborative filtering, neural network, content-based filtering, and knowledge-based filtering components.
    - Dense layer with linear activation function to produce a single output representing the predicted rating or score.

Layer	Output Shape	Param #	Connected to
xusers_in (InputLayer)	(None, 1)	0	[]
xproducts_in (InputLayer)	(None, 1)	0	[]
nn_xusers_emb (Embedding)	(None, 1, 50)	20138650	['xusers_in[0][0]']
nn_xproducts_emb (Embedding)	(None, 1, 50)	20138650	['xproducts_in[0][0]']
cf_xusers_emb (Embedding)	(None, 1, 50)	20138650	['xusers_in[0][0]']
cf_xproducts_emb (Embedding)	(None, 1, 50)	20138650	['xproducts_in[0][0]']
nn_xusers (Reshape)	(None, 50)	0	['nn_xusers_emb[0][0]']
nn_xproducts (Reshape)	(None, 50)	0	['nn_xproducts_emb[0][0]']
cf_xusers (Reshape)	(None, 50)	0	['cf_xusers_emb[0][0]']
cf_xproducts (Reshape)	(None, 50)	0	['cf_xproducts_emb[0][0]']
concatenate (Concatenate)	(None, 100)	0	['nn_xusers[0][0]', 'nn_xproducts[0][0]']
features_in (InputLayer)	(None, 35)	0	[]
contexts_in (InputLayer)	(None, 5079)	0	[]
cf_xx (Dot)	(None, 1)	0	['cf_xusers[0][0]', 'cf_xproducts[0][0]']
nn_xx (Dense)	(None, 25)	2525	['concatenate[0][0]']
features_x (Dense)	(None, 35)	1260	['features_in[0][0]']
context_x (Dense)	(None, 5079)	25801320	['contexts_in[0][0]']
concatenate_1 (Concatenate)	(None, 5140)	0	['cf_xx[0][0]', 'nn_xx[0][0]',

			'features_x[0][0]', 'context_x[0][0]']
y_out (Dense)	(None, 1)	5141	['concatenate_1[0][0]']
Total params: 106,364,846			
Trainable params: 106,364,846			
Non-trainable params: 0			

.

```
embeddings_size = 50
usr, prd = user.shape[0], procedures1.shape[0]
feat = len(features)
ctx = len(context1)

############################################ COLLABORATIVE FILTERING
#####################################################
# Input layer
xusers_in = layers.Input(name="xusers_in", shape=(1,))
xproducts_in = layers.Input(name="xproducts_in", shape=(1,))

# A) Matrix Factorization
## embeddings and reshape
cf_xusers_emb = layers.Embedding(name="cf_xusers_emb", input_dim=usr,
output_dim=embeddings_size)(xusers_in)
cf_xusers = layers.Reshape(name='cf_xusers',
target_shape=(embeddings_size,))(cf_xusers_emb)
## embeddings and reshape
cf_xproducts_emb = layers.Embedding(name="cf_xproducts_emb", input_dim=prd,
output_dim=embeddings_size)(xproducts_in)
cf_xproducts = layers.Reshape(name='cf_xproducts',
target_shape=(embeddings_size,))(cf_xproducts_emb)
## product
cf_xx = layers.Dot(name='cf_xx', normalize=True, axes=1)([cf_xusers,
cf_xproducts])

# B) Neural Network
## embeddings and reshape
nn_xusers_emb = layers.Embedding(name="nn_xusers_emb", input_dim=usr,
output_dim=embeddings_size)(xusers_in)
nn_xusers = layers.Reshape(name='nn_xusers',
target_shape=(embeddings_size,))(nn_xusers_emb)
## embeddings and reshape
nn_xproducts_emb = layers.Embedding(name="nn_xproducts_emb", input_dim=prd,
output_dim=embeddings_size)(xproducts_in)
nn_xproducts = layers.Reshape(name='nn_xproducts',
target_shape=(embeddings_size,))(nn_xproducts_emb)
```

```
## concat and dense
nn_xx = layers.Concatenate()([nn_xusers, nn_xproducts])
nn_xx = layers.Dense(name="nn_xx", units=int(embeddings_size/2),
activation='relu')(nn_xx)

############################################ CONTENT BASED
#####################################################
# Product Features
features_in = layers.Input(name="features_in", shape=(feat,))
features_x = layers.Dense(name="features_x", units=feat,
activation='relu')(features_in)

############################################ KNOWLEDGE BASED
#####################################################
# Context
contexts_in = layers.Input(name="contexts_in", shape=(ctx,))
context_x = layers.Dense(name="context_x", units=ctx,
activation='relu')(contexts_in)
```

4.7Combine the four algorithms

After build and designing the collaborative filtering (CF), content-based filtering (CBF), and knowledge-based filtering (KBF) models, we can now combine them to create a hybrid recommender system. The three models produce three sets of predictions, which we concatenate and pass through a dense layer to produce the final recommendation.

```
############################################ OUTPUT
#####################################################
# Merge all
y_out = layers.Concatenate()([cf_xx, nn_xx, features_x,
context_x])
y_out = layers.Dense(name="y_out", units=1,
activation='linear')(y_out)
```

4.8Compiled with the optimizier, loss function and metrics

After defining the neural network's architecture and selecting the appropriate activation functions, we compile the model using the Adam optimizer, mean absolute error loss function and mean absolute percentage error metrics .

```
# Compile
model = models.Model(inputs=[xusers_in,xproducts_in,
features_in, contexts_in], outputs=y_out, name="Hybrid_Model")
model.compile(optimizer='adam', loss='mean_absolute_error',
metrics=['mean_absolute_percentage_error'])
```

```
model.summary()
```

4.9Set the hyperparameter on the model

After compiling the model, we must set the hyperparameters used during training. Hyperparameters are parameters that are not learned during training but are set before the training process begins. In our case, we will set the number of epochs, the batch size, and the learning rate. We set the number of epochs to 120 and the batch size to 300. We also shuffle the training data during each epoch using the shuffle=True argument. We use a validation split of 15% using the validation_split=0.15 argument. Finally, we set verbose=1 to display the training progress during each epoch.

```
# Train the model using the TensorFlow tensors as input
training = model.fit(x=[train_df_1, train_df_2, train_df_3,
train_df_4], y=train_df_y, epochs=120,  batch_size=300 ,
shuffle=True, validation_split=0.15, verbose=1)
```

4.10 Train Artificial Neural Network

The training data is split into four numpy arrays representing the subject ID, procedure code, features, and context. The model is then trained using the fit method, and the training history is stored in the 'training' variable. The training history can be used to visualize the training and validation loss over epochs and help diagnose issues such as overfitting or underfitting.

To evaluate the performance of the model during training, we set verbose to 1 so that the training process will be displayed and we can track our progress of training on each epoch.

```
Epoch 1/120
913/913 [==============================] - 103s 109ms/step -
loss: 1.5911 - mean_absolute_percentage_error: 51.7103 -
val_loss: 1.4268 - val_mean_absolute_percentage_error:
47.3203
Epoch 2/120
913/913 [==============================] - 99s 108ms/step -
loss: 1.1434 - mean_absolute_percentage_error: 36.6165 -
val_loss: 1.2798 - val_mean_absolute_percentage_error:
46.1389
Epoch 3/120
913/913 [==============================] - 94s 103ms/step -
loss: 0.9528 - mean_absolute_percentage_error: 31.1705 -
val_loss: 1.2121 - val_mean_absolute_percentage_error:
46.3791
Epoch 4/120
913/913 [==============================] - 94s 103ms/step -
loss: 0.8419 - mean_absolute_percentage_error: 27.5473 -
val_loss: 1.1596 - val_mean_absolute_percentage_error:
44.8981
```

```
Epoch 5/120
913/913 [==============================] - 96s 106ms/step -
loss: 0.7716 - mean_absolute_percentage_error: 25.8006 -
val_loss: 1.1335 - val_mean_absolute_percentage_error:
45.0710
.....................................................
Epoch 116/120
913/913 [==============================] - 14s 15ms/step -
loss: 0.2779 - mean_absolute_percentage_error: 9.7627 -
val_loss: 0.9778 - val_mean_absolute_percentage_error:
44.3811
Epoch 117/120
913/913 [==============================] - 14s 15ms/step -
loss: 0.2774 - mean_absolute_percentage_error: 9.7557 -
val_loss: 0.9760 - val_mean_absolute_percentage_error:
44.1537
Epoch 118/120
913/913 [==============================] - 14s 15ms/step -
loss: 0.2765 - mean_absolute_percentage_error: 9.6967 -
val_loss: 0.9768 - val_mean_absolute_percentage_error:
44.3932
Epoch 119/120
913/913 [==============================] - 14s 15ms/step -
loss: 0.2762 - mean_absolute_percentage_error: 9.6972 -
val_loss: 0.9742 - val_mean_absolute_percentage_error:
44.1649
Epoch 120/120
913/913 [==============================] - 14s 15ms/step -
loss: 0.2756 - mean_absolute_percentage_error: 9.6736 -
val_loss: 0.9774 - val_mean_absolute_percentage_error:
44.3226
```

4.11 Generate 10 recommendation

Based on a trained machine-learning model, the model generates recommendations for a given user. The model inputs the user's context and features information, such as demographic, medical history, and features related to medical procedures. It then predicts the outcome of these procedures and recommends the top 10 procedures with the highest predicted outcomes.

We drops duplicate rows of the procedures dataset based on the "icd_code_P_1" column and resets the index. The resulting dataset is assigned to "procedures1". We use the "get_user_context" function inputs the user ID, the complete dataset, and the user's context information. It returns a subset of the complete dataset that contains only the rows corresponding to the user and the context information. Then we create "create_procedures_df" function takes the user ID and the procedures dataset as inputs and creates a new dataset with the user ID added as a new column.

The "generate_recommendations" function takes the user ID, the trained model, the procedures dataset, the complete dataset, the features related to medical procedures, and the user's context information as inputs. It first calls the "get_user_context" function to obtain the user context information and then calls the "create_procedures_df"

function to create a new dataset with the user ID added. The two datasets are merged based on the "subject_id_1" column, and the resulting dataset is converted to a Pandas DataFrame. The model is then applied to the dataset, and the predicted outcomes are added as a new column called "yhat". The top 10 procedures with the highest predicted outcomes are then selected and returned as a Pandas DataFrame.

The code then chooses a user with input ID from the dataset and displays the user's medical history and procedures records. It then calls the "generate_recommendations" function to generate recommendations for the user and displays the top 10 recommended procedures along with their predicted outcomes.

```
from pandas.io.formats.style import Subset
def get_user_context(user_id, complete_df, context1):
    context1 = [col for col in context1 if col in complete_df.columns]
    return complete_df.loc[complete_df["subject_id_1"]==user_id,
["subject_id_1"] + context1]

def create_procedures_df(user_id, procedures1):
    procedures1 = procedures1.drop_duplicates().reset_index(drop=True)
    procedures1['subject_id_1'] = user_id
    return procedures1


def generate_recommendations(user_id, model, procedures1, complete_df,
features, context1):
    user_context = get_user_context(user_id, complete_df, context1)
    procedures1 = create_procedures_df(user_id, procedures1)
    procedures1 = cudf.merge(procedures1, user_context,
on='subject_id_1', how='left')
    procedures1 = procedures1.to_pandas()

    procedures1["yhat"] = model.predict([procedures1["subject_id_1"],
procedures1["icd_code_P_1"], procedures1[features],
procedures1[context1]])
    return procedures1.sort_values("yhat",
ascending=False).drop_duplicates(subset='icd_code_P_1').nlargest(10,
'yhat')

# Choose a random user
user_id = 50
user_data = sampled_data.loc[sampled_data['subject_id_1'] == user_id]

# Generate recommendations for the user
rekomendasiX6 = generate_recommendations(user_id, model, procedures1,
complete_df, features, context1)

display(user_data[['long_title','seq_num','icd_code','icd_version']])
```

```
print("Hey, it seems you have been diagnosed with:")
print(user_data['long_title'].unique().to_pandas().squeeze())
print("and also have procedures records for:")
print(user_data['long_title_P'].unique().to_pandas().squeeze())
print("We have some interesting procedures that may be of interest to
you.")
display(rekomendasiX6[['long_title_P', 'yhat']])
```

```
910/910 [==============================] - 2s 2ms/step
```

	long_title	seq_num	icd_code	icd_version
189272	Encounter for palliative care	21	V667	9
196163	Hyperpotassemia	13	2767	9
363316	Cardiac pacemaker in situ	38	V4501	9

```
Hey, it seems you have been diagnosed with:
0        Cardiac pacemaker in situ
1    Encounter for palliative care
2                  Hyperpotassemia
Name: long_title, dtype: object
and also have procedures records for:
0      Pulmonary artery wedge monitoring
1    Right heart cardiac catheterization
2                           Thoracentesis
Name: long_title_P, dtype: object
We have some interesting procedures that may be of interest to you.
```

	long_title_P	yhat
4833	Drainage of Right Middle Lobe Bronchus, Via Na...	11.132213
6957	Inspection of Upper Extremity Subcutaneous Tis...	9.361391
21696	Change Pressure Dressing on Left Upper Leg	9.096400
2316	Removal of Tunneled Vascular Access Device fro...	8.902567
3405	Introduction of Destructive Agent into Periphe...	8.156526
11550	Change Drainage Device in Peritoneal Cavity, E...	8.058792
2139	Introduction of Other Anti-infective into Eye,...	7.445644
16491	Bypass Coronary Artery, Two Arteries from Righ...	7.388990
14883	Control Bleeding in Abdominal Wall, Open Approach	7.376282
19665	Fusion or refusion of 2-3 vertebrae	7.302238

Chapter V

System Testing and Discussion

5.1 Presentation and discussion of the system testing results

5.1.1 Training history

We evaluate the performance of our recommender system during the training process. The model's performance during training was evaluated using a validation dataset with a validation_split of 0.15. We generate four plots to evaluate the performance of the model during training. The first two plots show the model's mean absolute percentage error (MAPE), one for the training set and one for the validation set, over all epochs. The third plot shows the loss value of the model (in this case, mean absolute error), also for both the training and validation sets. The final plot shows only the validation loss.

Epoch	Training Loss	Training MAPE	Validation Loss	Validation MAPE
1	15.911	517.103	14.268	473.203
2	11.434	366.165	12.798	461.389
3	0.9528	311.705	12.121	463.791
4	0.8419	275.473	11.596	448.981
5	0.7716	258.006	11.335	450.710
6	0.7148	237.707	11.044	439.510
7	0.6763	228.613	10.850	434.732
8	0.6407	215.708	10.757	443.471
9	0.6155	210.052	10.575	435.240
10	0.5901	200.144	10.497	435.010
11	0.5718	196.806	10.416	432.469

Epoch	Training Loss	Training MAPE	Validation Loss	Validation MAPE
61	0.3320	116.924	0.9764	436.934
62	0.3309	116.303	0.9760	437.083
63	0.3290	115.819	0.9772	437.728
64	0.3275	115.238	0.9760	436.100
65	0.3261	114.851	0.9785	440.298
66	0.3251	114.391	0.9778	439.036
67	0.3236	113.900	0.9742	435.260
68	0.3222	113.479	0.9793	442.012
69	0.3211	113.000	0.9783	441.144
70	0.3197	112.697	0.9773	439.726
71	0.3182	111.909	0.9763	438.311

12	0.5528	188.755	10.403	440.364
13	0.5387	185.825	10.264	428.379
14	0.5231	179.817	10.274	434.007
15	0.5126	177.157	10.229	433.959
16	0.5005	172.516	10.167	431.352
17	0.4910	170.260	10.117	429.524
18	0.4816	166.742	10.101	431.986
19	0.4739	164.427	10.094	433.786
20	0.4646	160.887	10.080	434.576
21	0.4590	159.829	10.075	436.519
22	0.4510	156.842	10.019	432.708
23	0.4453	155.006	0.9991	428.707
24	0.4385	152.758	0.9986	434.370
25	0.4330	151.206	0.9978	431.229
26	0.4277	149.159	0.9929	431.112
27	0.4233	148.008	0.9905	429.399
28	0.4177	145.793	0.9908	430.731
29	0.4142	145.019	0.9976	440.796
30	0.4088	143.130	0.9944	438.314
31	0.4061	142.260	0.9871	430.228

72	0.3171	111.537	0.9744	435.896
73	0.3156	111.051	0.9753	435.774
74	0.3148	110.754	0.9786	441.851
75	0.3132	110.198	0.9744	437.495
76	0.3126	109.964	0.9746	437.855
77	0.3112	109.649	0.9730	437.362
78	0.3103	109.231	0.9775	441.043
79	0.3085	108.644	0.9732	438.828
80	0.3079	108.194	0.9737	436.257
81	0.3065	107.817	0.9761	439.104
82	0.3059	107.655	0.9769	439.811
83	0.3047	107.322	0.9758	441.061
84	0.3039	106.920	0.9763	440.823
85	0.3028	106.499	0.9759	439.372
86	0.3025	106.375	0.9775	438.555
87	0.3005	105.747	0.9745	438.653
88	0.3001	105.660	0.9747	438.764
89	0.2987	105.039	0.9757	438.593
90	0.2980	104.763	0.9791	443.721
91	0.2968	104.405	0.9767	441.692

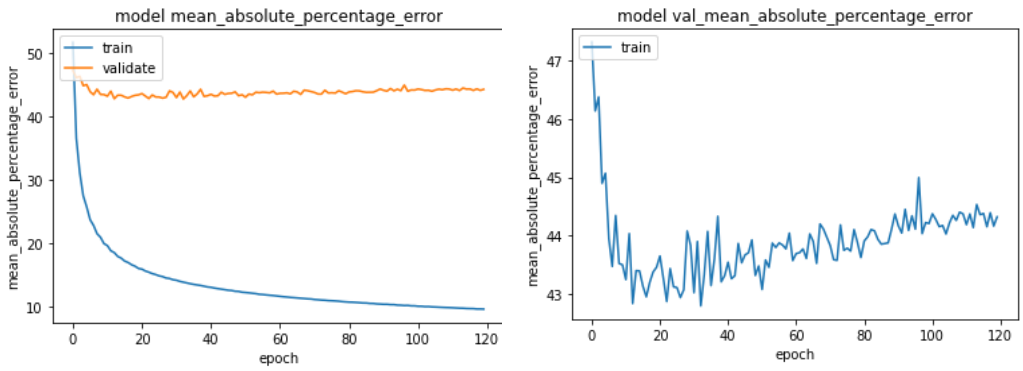
32	0.4016	140.524	0.9951	439.009
33	0.3978	139.236	0.9818	427.975
34	0.3938	138.079	0.9867	433.279
35	0.3910	136.856	0.9936	440.742
36	0.3872	135.772	0.9843	431.452
37	0.3847	134.878	0.9866	436.030
38	0.3816	134.056	0.9931	443.327
39	0.3783	132.882	0.9797	432.098
40	0.3753	131.726	0.9819	433.147
41	0.3731	131.124	0.9833	435.453
42	0.3703	129.988	0.9793	432.624
43	0.3677	129.075	0.9801	433.134
44	0.3656	128.267	0.9828	438.675
45	0.3627	127.433	0.9806	435.360
46	0.3603	126.510	0.9797	436.719
47	0.3587	126.082	0.9808	437.071
48	0.3558	124.940	0.9824	439.282
49	0.3540	124.688	0.9762	433.190
50	0.3514	123.455	0.9779	434.804
51	0.3498	122.973	0.9742	430.780

92	0.2969	104.397	0.9754	440.459
93	0.2952	103.872	0.9789	444.512
94	0.2949	103.846	0.9757	440.906
95	0.2934	103.071	0.9773	443.409
96	0.2930	102.895	0.9756	441.118
97	0.2916	102.470	0.9844	449.970
98	0.2915	102.620	0.9783	440.358
99	0.2902	101.972	0.9759	442.252
100	0.2895	101.928	0.9775	442.085
101	0.2888	101.301	0.9768	443.753
102	0.2879	101.227	0.9767	442.785
103	0.2868	100.712	0.9759	441.551
104	0.2867	100.782	0.9756	441.748
105	0.2856	100.302	0.9733	440.297
106	0.2848	100.087	0.9761	442.169
107	0.2840	99.788	0.9760	443.498
108	0.2835	99.532	0.9779	442.618
109	0.2826	99.298	0.9778	444.034
110	0.2821	99.132	0.9792	443.732
111	0.2816	98.902	0.9749	441.893

52	0.3482	122.437	0.9777	435.837	112	0.2805	98.602	0.9785	443.723
53	0.3459	121.775	0.9772	434.574	113	0.2802	98.308	0.9751	441.395
54	0.3444	121.180	0.9803	438.723	114	0.2795	98.170	0.9801	445.341
55	0.3418	120.128	0.9791	437.978	115	0.2784	97.778	0.9773	443.608
56	0.3405	119.761	0.9798	438.760	116	0.2779	97.627	0.9778	443.811
57	0.3388	119.162	0.9795	438.402	117	0.2774	97.557	0.9760	441.537
58	0.3370	118.625	0.9773	437.742	118	0.2765	96.967	0.9768	443.932
59	0.3357	118.172	0.9808	440.439	119	0.2762	96.972	0.9742	441.649
60	0.3341	117.588	0.9731	435.736	120	0.2756	96.736	0.9774	443.226

The results of training the model show that the model's performance improved significantly throughout training. The MAPE for the training set decreased from 517.103 in the first epoch to 96.736 in the 120th epoch, while the MAPE for the validation set decreased from 473.203 to 443.226 over the same period. The loss value of the model decreased from 15.911 in the first epoch to 0.2756 in the 120th epoch, while the validation loss decreased from 14.268 to 0.9774.

We have presented the results of our training process in. The table lists each epoch's training loss, training MAPE, validation loss, and validation MAPE. The training loss and MAPE show a decreasing trend, indicating that the model is learning and improving over time. The validation loss and MAPE also show a decreasing trend up to a certain point, after which they stabilize or start to increase. This indicates that the model may be overfitting to the training data and may not generalize well to new data.



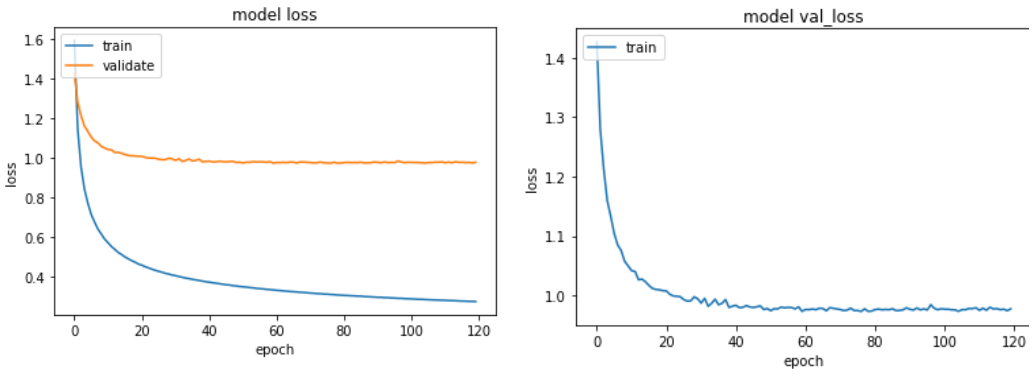


Figure 31 training history plot

.

5.1.2 Evaluation metrics

5.1.2.1 MAE

The mean absolute error (MAE) measures the average absolute difference between the predicted and actual values. In your case, the MAE of 1 indicates that on average, the predicted outcome differs from the actual outcome by 1 unit. This means that your model's predictions are, on average, off by 1 unit. A low MAE value indicates that the model predicts the actual values well. Therefore, the low MAE value of 1 indicates that your model makes accurate predictions.

```
# overall
y_test = test_df1["y"]
predicted = test_df1["yhat"]

print("Mean Absolute Error (Σ|y-pred|/n):",
"{:,.0f}".format(metrics.mean_absolute_error(y_test, predicted)))
```

5.1.2.2 MAPE

The mean absolute percentage error (MAPE) measures the average absolute percentage difference between the predicted and actual values. In your case, the MAPE of 0.44% indicates that on average, the predicted outcome differs from the actual outcome by 0.44% of the actual value. This means that your model's predictions are, on average, off by 0.44% of the actual value. A low MAPE value indicates that your model performs well in predicting the actual values. Therefore, the low MAPE value of 0.44% indicates that your model makes accurate predictions.

```
print("Mean Absolute Perc Error (Σ(|y-pred|/y)/n):",
str(round(np.mean(np.abs((y_test-predicted)/predicted)), 2))+"%")
```

5.2Generate 10 procedures

We generate 10 procedures for several patients and it works by making recommended procedures for each patient. It seems personalized and relevant to their medical condition, showing the potential of using recommender systems in healthcare. By analyzing a patient's medical history and procedures, the system can provide recommendations that could relate with their health history.

After generating a recommendation for 7 patients, there seems to be a high frequency of recommendations related to inspections, drainages, and removals of devices. This could be because these procedures are commonly performed in hospitals and have a higher chance of being recommended. This also highlights the need for more diverse and innovative procedures to be included in the system to provide patients with a broader range of options. The varying levels of frequency for each recommendation suggest that some procedures are more common than others, which could be due to differences in medical conditions and patient preferences

Procedure	Frequency
Inspection of Upper Extremity Subcutaneous Tissue	5
Drainage of Right Middle Lobe Bronchus, Via Natural or Artificial Opening	4
Introduction of Other Therapeutic Substance into Subcutaneous Tissue and Fascia, Percutaneous Approach	3
Change Drainage Device in Peritoneal Cavity, External Approach	3
Removal of Internal Fixation Device from Right Upper Arm Bone, Open Approach	3
Rhinoscopy	2
Fluoroscopy of Dialysis Shunt/Fistula using Other Contrast	2
Introduction of Destructive Agent into Peripheral Nerve, Percutaneous Approach	2
Removal of Other Device on Chest Wall, External Approach	2
Change Pressure Dressing on Left Upper Leg, External Approach	2
Drainage of Bladder with Drainage Device, Via Natural or Artificial Opening	2
Dilation of Esophagus, Via Natural or Artificial Opening	2
Inspection of Tracheobronchial Tree, Via Natural or Artificial Opening	1
Extirpation of Matter from Stomach, Via Natural or Artificial Opening	1
Dilation of Common Bile Duct, Percutaneous Approach	1
Drainage of Right Kidney Pelvis, Percutaneous Approach	1
Removal of Tunneled Vascular Access Device from Upper Arm, External Approach	1
Beam Radiation of Abdomen using Photons 1 - 10 MeV, External Approach	1
Injection of anesthetic into spinal canal for other than surgery, diagnostic or therapeutic purposes, lumbar	1
Drainage of Right Kidney Pelvis, Percutaneous Approach	1
Fluoroscopy of Dialysis Shunt/Fistula using Other Contrast	1
Removal of Tracheostomy Device from Trachea, External Approach	1

Drainage of Left Main Bronchus, Via Natural or Artificial Opening	1
Transfer of Facial Muscle, Open Approach	1
Insertion of three vascular stents, Percutaneous Approach	1
Monitoring of Cardiac Output, Percutaneous Approach	1
Procedure on single vessel, Percutaneous Approach	1
Excision of Right Upper Arm Skin, External Approach	1
Introduction of Other Anti-infective into Eye, Percutaneous Approach	1
Endoscopic destruction of other lesion or tissue of head and neck	1
Extirpation of Matter from Right Lower Lobe Bronchus, Via Natural or Artificial Opening	1

The lack of information about the reliability and validity of the recommendations provided by the recommender system. While the system could accurately predict a patient's most relevant clinical trial, we could not assess whether the recommendations made sense to the physician or were perceived as useful in practice.

Chapter VI

Conclusion and Future Work

6.1Summary of the study and main findings

The development of a hybrid recommender system for clinical decision support systems has the potential to significantly improve the quality of patient care and enhance hospital performance. This project explored the benefits of combining different recommendation algorithms to provide more accurate and reliable recommendations that benefit doctors, hospitals, and patients. The results of the training process of our system were presented, and despite the limitations related to data availability and quality, user acceptance, legal and ethical considerations, limited generalizability, and dependence on technology, the potential benefits of this system are substantial.

Our research has shown that this hybrid recommender system can generate procedures on patients that the hospital can use to provide optional clinical decisions to healthcare professionals and engage patients in their treatment decisions, resulting in improved patient satisfaction. Moreover, the system hopefully contributes to clinical decision support systems and provides a useful tool for healthcare professionals to make more informed and accurate decisions about patient care. The personalized recommendations this system provides also help patients understand their treatment options and make informed decisions about their care. Therefore, the successful implementation of this system can lead to improved hospital performance and patient outcomes.

1. The development of a hybrid recommender system for clinical decision support systems has the potential to significantly improve the quality of patient care and enhance hospital performance.
2. The hybrid recommender system can provide accurate and reliable recommendations that benefit doctors, hospitals, and patients, improving patient satisfaction.
3. The personalized recommendations this system provides also help patients understand their treatment options and make informed decisions about their care.
4. The presented training results show that the hybrid recommender system performed well on the training and validation sets, with a mean absolute percentage error (MAPE) of less than 5% and a low loss value.
5. The system's limitations include dependence on data availability and quality, user acceptance, legal and ethical considerations, limited generalizability, and dependence on technology.
6. The system is a valuable tool for healthcare professionals and patients to make more informed decisions about patient care.
7. The model's performance improved significantly throughout training, as evidenced by the decreasing trend in the MAPE and loss value over the epochs.

8. The results of testing the model on a test dataset show that the system has a low mean absolute error (MAE) and mean absolute percentage error (MAPE), indicating good model performance.

9. The system generated 10 procedures on several patients using the recommender system, which showed promising results in recommending procedures for each patient.

6.2Future recommendations for improving the hybrid recommender system and its potential applications

1. Future research in this field should focus on a tailored recommendation system since our recommendation system recommends the top 10 procedures. The top 10 recommendations have no sequential order of tailored treatment for the patient. This can be done by incorporating attention-based models in the hybrid recommender system to improve the recommendations' accuracy further.

2. Future studies should investigate ways to address the limitations of data availability and quality, user acceptance, legal and ethical considerations, limited generalizability, and dependence on technology through collaborations between healthcare providers, researchers, and technology experts.

3. Healthcare providers should consider integrating hybrid recommender systems in their clinical decision-making processes to enhance the quality of patient care and improve hospital performance.

4. Pilot studies and evaluations of the system's effectiveness and impact on patient outcomes should be conducted to ensure successful implementation.

5. The development and implementation of hybrid recommender systems in clinical decision support systems have the potential to revolutionize patient care and lead to improved healthcare outcomes.

6.3Conclusion and final remarks

In summary, developing a hybrid recommender system for clinical decision support has shown promising results in improving hospital performance and benefiting doctors, hospitals, and patients. This personalized recommendations system and help patients to make informed decisions about their care, resulting in improved patient satisfaction. The system is a valuable tool for healthcare professionals to make more informed decisions about patient care. However, there are limitations related to data availability and quality, user acceptance, legal and ethical considerations, limited generalizability, and dependence on technology that need to be addressed.

Future research should focus on tailored recommendation systems, incorporating attention-based models in the hybrid recommender system to improve the accuracy of recommendations further. Collaboration between healthcare providers, researchers, and technology experts is needed to address the limitations of data availability and quality, user acceptance, legal and ethical considerations, limited generalizability, and dependence on technology.

Healthcare providers should consider integrating hybrid recommender systems in their clinical decision-making processes to enhance the quality of patient care and improve hospital performance. Pilot studies and evaluations of the system's effectiveness and impact on patient outcomes should be conducted to ensure successful implementation. The development and implementation of hybrid recommender systems in clinical decision support systems have the potential to revolutionize patient care and lead to improved healthcare outcomes.

Reference:

Aggarwal, C. C. (2016). Ensemble-Based and Hybrid Recommender Systems. In *Recommender Systems* (pp. 199–224). Springer, Cham. https://doi.org/10.1007/978-3-319-29659-3_6

Aggarwal, C. C. (2018). An Introduction to Neural Networks. In *Neural Networks and Deep Learning* (pp. 1–52). Springer, Cham. https://doi.org/10.1007/978-3-319-94463-0_1

Bankhele, S., Mhaske, A., Bhat, S., & Shinde, S. V. (2017). A Diabetic Healthcare Recommendation System. *International Journal of Computer Applications*, *167*(5), 14–18. https://doi.org/10.5120/IJCA2017914246

Bokde, D., Girase, S., & Mukhopadhyay, D. (2015). Matrix Factorization Model in Collaborative Filtering Algorithms: A Survey. *Procedia Computer Science*, *49*(1), 136–146. https://doi.org/10.1016/J.PROCS.2015.04.237

cakir, M., oguducu, sule gunduz, & tugay, resul. (2020). *A Deep Hybrid Model for Recommendation Systems*. https://doi.org/10.48550/arxiv.2009.09748

Croskerry, P. (2009). Clinical cognition and diagnostic error: Applications of a dual process model of reasoning. *Advances in Health Sciences Education*, *14*(1 SUPPL), 27–35. https://doi.org/10.1007/S10459-009-9182-2/METRICS

De Myttenaere, A., Golden, B., Grand, B. Le, & Rossi, F. (2016). Mean Absolute Percentage Error for regression models. *Neurocomputing*, *192*, 38–48. https://doi.org/10.1016/j.neucom.2015.12.114

Gräßer, F., Tesch, F., Schmitt, J., Abraham, S., Malberg, H., & Zaunseder, S. (2021). A pharmaceutical therapy recommender system enabling shared decision-making. *User Modeling and User-Adapted Interaction*, *32*(5), 1019–1062. https://doi.org/10.1007/S11257-021-09298-4/FIGURES/14

Gündoğan, E., & Kaya, M. (2022). A novel hybrid paper recommendation system using deep learning. *Scientometrics*, *127*(7), 3837–3855. https://doi.org/10.1007/S11192-022-04420-8/METRICS

Han, Q., Ji, M., Martinez De Rituerto De Troya, I., Gaur, M., & Zejnilovic, L. (2018). A Hybrid Recommender System for Patient-Doctor Matchmaking in Primary Care. *Proceedings - 2018 IEEE 5th International Conference on Data Science and Advanced Analytics, DSAA 2018*, 481–490. https://doi.org/10.1109/DSAA.2018.00062

Hancock, J. T., & Khoshgoftaar, T. M. (2020). Survey on categorical data for neural networks. *Journal of Big Data*, *7*(1), 1–41. https://doi.org/10.1186/S40537-020-00305-W/FIGURES/4

He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T. S. (2017). Neural Collaborative Filtering. *26th International World Wide Web Conference, WWW 2017*, 173–182.

https://doi.org/10.48550/arxiv.1708.05031

Ian Goodfellow, Y. B. and A. C. (2016). Optimization for Training Deep Models. In *Deep Learning* (pp. 274–317). The MIT Press. deeplearningbook.org

Jones, C., Thornton, J., & Wyatt, J. C. (2021). Enhancing trust in clinical decision support systems: a framework for developers. *BMJ Health & Care Informatics*, *28*(1), e100247. https://doi.org/10.1136/BMJHCI-2020-100247

Kingma, D. P., & Ba, J. L. (2014). Adam: A Method for Stochastic Optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. https://doi.org/10.48550/arxiv.1412.6980

MIMIC-IV v2.1. (n.d.). Retrieved December 23, 2022, from https://physionet.org/content/mimiciv/2.1/

Nasiri, M., Minaei, B., & Kiani, A. (2016). Dynamic Recommendation: Disease Prediction and Prevention Using Recommender System. *International Journal of Basic Science in Medicine*, *1*(1), 13–17. https://doi.org/10.15171/IJBSM.2016.04

Sutton, R. T., Pincock, D., Baumgart, D. C., Sadowski, D. C., Fedorak, R. N., & Kroeker, K. I. (2020). An overview of clinical decision support systems: benefits, risks, and strategies for success. *NPJ Digital Medicine*, *3*(1). https://doi.org/10.1038/S41746-020-0221-Y

Willmott, C. J., & Matsuura, K. (2005). Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. *Climate Research*, *30*(1), 79–82. https://doi.org/10.3354/CR030079

Zhang, Q., Zhang, G., Lu, J., & Wu, D. (2015). A Framework of Hybrid Recommender System for Personalized Clinical Prescription. *2015 10th International Conference on Intelligent Systems and Knowledge Engineering (ISKE)*, 189–195. https://doi.org/10.1109/ISKE.2015.98

Appendix: Additional Technical Details

1.1. Survey at Jogja International Hospital

	Timestamp	45317,42409
	Harap berikan informasi berikut untuk membantu kami memahami peran Anda dalam organisasi. Jawaban Anda akan tetap rahasia. Nama Lengkap:	arbaati
	Departemen/Tim:	Pemasaran dan Penjualan
	Posisi/Peran:	Manager Pemasaran dan Penjualan
	Informasi Kontak (Opsional): Email: telp:	082138237332
	Mohon beri tahu kami tingkat keterlibatan Anda dengan promosi layanan kesehatan: Tingkat Keterlibatan:	Terlibat langsung dalam keputusan promosi layanan kesehatan
	Memahami Tujuan Tim Pemasaran: Kami ingin mengetahui dan mengIdentifikasi tujuan dan sasaran utama tim pemasaran rumah sakit dalam mempromosikan layanan kesehatan.	
1	Apa fokus utama dari upaya promosi tim pemasaran rumah sakit Anda?	Meningkatkan kesadaran Merk (Positioning) , Mempromosikan Produk dan Layanan dll, Meningkatkan kunjungan Pasien
2	Tujuan spesifik apa yang diprioritaskan tim pemasaran Anda dalam promosi layanan kesehatan?	Pasien baru, Market retention , kepuasan pelanggan
3	Bagaimana tim pemasaran Anda mengukur keberhasilan kampanye promosi?	Resume Survey Kepuasan Pelanggan, Kunjungan pasien baru dan lama
4	Dalam konteks promosi layanan kesehatan, tujuan jangka panjang apa yang ingin dicapai oleh tim pemasaran Anda?	Sebagai RS Rujukan Terbaik DIY-Jateng , Menetapkan RS sebagai market leader area Regional
5	Bagaimana tim pemasaran Anda saat ini memprioritaskan antara layanan kesehatan yang berbeda ketika merencanakan kegiatan promosi?	Berdasarkan Strategis RS, RKAP dan Tren Penyakit atau layanan baru
	Menilai Tantangan Saat Ini: Kami ingin mengetahui tantangan yang dihadapi oleh tim pemasaran dalam pengambilan keputusan dan upaya promosi mereka saat ini.	
6	Apa hambatan atau tantangan utama yang dihadapi tim pemasaran Anda dalam mempromosikan layanan kesehatan secara efektif?	Bersaing dengan penyedia layanan kesehatan lainnya

7	Bagaimana Anda menilai efektivitas saluran komunikasi Anda saat ini dalam menjangkau dan terlibat dengan audiens target Anda?	Sangat efektif
8	Apa tantangan utama yang dihadapi tim pemasaran Anda dalam mengukur dampak kampanye promosi?	Kurangnya alat untuk analisis kampanye real-time
9	Bagaimana tim pemasaran Anda saat ini beradaptasi dengan perubahan dalam lanskap perawatan kesehatan atau menanggapi tren yang muncul di industri?	Pemantauan proaktif dan penyesuaian strategi
10	Menurut Anda, apa hambatan terbesar untuk kolaborasi yang efektif antara tim pemasaran dan departemen lain di dalam rumah sakit?	Saat ini Tidak ada kendala
	Proses Pengambilan Keputusan: Kami ingin mendapatkan wawasan tentang bagaimana tim pemasaran saat ini membuat keputusan mengenai promosi layanan dan iklan.	
11	Faktor atau kriteria apa yang dipertimbangkan tim pemasaran Anda ketika memutuskan layanan kesehatan untuk dipromosikan?	Wawasan dan analitik berbasis data
12	Bagaimana tim pemasaran Anda mengumpulkan informasi untuk menginformasikan pengambilan keputusan dalam promosi layanan?	Riset pasar dan analisis kompetitif
13	Peran apa yang dimainkan umpan balik pasien dalam mempengaruhi keputusan tim pemasaran tentang promosi layanan dan iklan?	Dipertimbangkan tetapi bukan faktor utama
14	Jelaskan proses khas yang diikuti tim pemasaran Anda mulai dari mengidentifikasi layanan kesehatan untuk promosi hingga menjalankan kampanye pemasaran.	Perencanaan strategis dan penetapan tujuan
15	Bagaimana tim pemasaran Anda menilai keberhasilan atau efektivitas kampanye promosi setelah dijalankan?	Meninjau angka perolehan pasien
	Jelajahi Preferensi Informasi: Kami ingin memahami jenis informasi dan rekomendasi tim pemasaran menemukan paling berharga dalam kegiatan promosi mereka.	
16	Jenis informasi apa yang menurut Anda paling berharga saat membuat konten promosi untuk layanan kesehatan?	Edukasi Layanan oleh dokter yang kompeten, Video layanan, Testimoni pasien dll
17	Bagaimana tim pemasaran Anda saat ini menggunakan data dan analitik untuk menginformasikan strategi promosi?	Pemanfaatan data Customer Relation Managemen (CRM)
18	Dalam mempromosikan layanan kesehatan, peran apa yang dimainkan oleh informasi berbasis bukti dan penelitian medis dalam pengambilan keputusan tim pemasaran Anda?	Peran penting dalam membentuk konten dan pesan
19	Jenis rekomendasi atau wawasan apa yang menurut tim pemasaran Anda paling bermanfaat dalam mengoptimalkan kegiatan promosi untuk layanan kesehatan?	Wawasan tentang saluran komunikasi yang efektif
	Periksa Persyaratan Sistem: Kami ingin mengidentifikasi fitur atau fungsi spesifik apa pun yang ingin dilihat oleh tim pemasaran dalam sistem rekomendasi yang dirancang untuk promosi layanan kesehatan.	
20	Fitur spesifik apa yang Anda yakini penting bagi sistem pemberi rekomendasi untuk secara efektif mendukung upaya promosi layanan kesehatan Anda?	Algoritme rekomendasi yang dipersonalisasi
21	Seberapa penting kemampuan untuk menyesuaikan dan menyesuaikan rekomendasi berdasarkan kebutuhan dan karakteristik unik audiens target Anda?	Sangat penting
22	Tingkat integrasi apa dengan sistem atau basis data rumah sakit lain yang Anda bayangkan untuk sistem pemberi rekomendasi?	Custamer Relationship Manajemen (CRM)
23	Apakah ada tantangan atau pain point khusus dalam proses promosi Anda saat ini yang Anda yakini dapat diatasi oleh sistem pemberi rekomendasi?	Tantangan dalam mempersonalisasi konten untuk demografi pasien yang berbeda

24	Email Address	marketingjih@gmail.com

1.2. Additional technical information on the deep learning models, hyperparameters, and optimization techniques used in building the hybrid recommender system.

```
### (master data and sampled data)
"""

import cudf, cuml, cupy

admissions = cudf.read_csv('/content/drive/MyDrive/MIMIC-IV
v2.2/admissions.csv', usecols=['subject_id', 'hadm_id', 'admission_type',
'admission_location', 'insurance', 'marital_status', 'race'])
patients = cudf.read_csv('/content/drive/MyDrive/MIMIC-IV v2.2/patients.csv',
usecols=['subject_id','gender', 'anchor_age'])
diagnoses_icd = cudf.read_csv('/content/drive/MyDrive/MIMIC-IV
v2.2/diagnoses_icd.csv', usecols=['subject_id', 'hadm_id','icd_code',
'icd_version', 'seq_num'])
d_icd_diagnoses = cudf.read_csv('/content/drive/MyDrive/MIMIC-IV
v2.2/d_icd_diagnoses.csv', usecols=['icd_code','icd_version','long_title'])
services = cudf.read_csv('/content/drive/MyDrive/MIMIC-IV v2.2/services.csv',
usecols=['subject_id', 'hadm_id','prev_service', 'curr_service'])
procedures_icd = cudf.read_csv('/content/drive/MyDrive/MIMIC-IV
v2.2/procedures_icd.csv', usecols=['subject_id', 'hadm_id','icd_code',
'icd_version', 'seq_num'])
d_icd_procedures = cudf.read_csv('/content/drive/MyDrive/MIMIC-IV
v2.2/d_icd_procedures.csv', usecols=['icd_code','icd_version','long_title'])

demografi = admissions.merge(patients, on='subject_id')
diagnoses = diagnoses_icd.merge(d_icd_diagnoses,
on=['icd_code','icd_version'])
procedures = procedures_icd.merge(d_icd_procedures,
on=['icd_code','icd_version'])
procedures = procedures.rename(columns={'icd_code': 'icd_code_P',
'icd_version': 'icd_version_P', 'seq_num': 'seq_num_P','long_title':
'long_title_P'})
diagnose = diagnoses.merge(demografi, on=['subject_id', 'hadm_id'])
diagnose = diagnose.merge(services, on=['subject_id', 'hadm_id'])
diagnose = diagnose.fillna(0)
diagnose.set_index(['subject_id','hadm_id'], inplace=True)
procedures.set_index(['subject_id','hadm_id'], inplace=True)
```

```
master_data = diagnose.join(procedures, how='outer').reset_index()
master_data = master_data.dropna()
#print(procedures.isna().sum())
#print(diagnose.isna().sum())
print(master_data.isna().sum())
print(master_data.shape)

master_data.to_parquet('/content/drive/MyDrive/123/rapids
jormungandr/master_data.parquet',compression='snappy')

print(master_data.shape)
print(master_data.icd_code.value_counts().shape)
print(master_data.icd_code_P.value_counts().shape)

import matplotlib.pyplot as plt

fig, axs = plt.subplots(3, 1, figsize=(20,10), sharey=True)

master_data.icd_code_P.value_counts().to_pandas().plot.bar(ax=axs[0],color='gr
een')
axs[0].set_title("icd_code_P by user master data")
axs[0].grid(axis='y')
axs[0].set_xlabel('procedures code')
axs[0].set_xticklabels([])

master_data.icd_code_P.value_counts().to_pandas().head(300).plot.bar(ax=axs[1]
,color='green')
axs[1].set_title("300 icd_code_P by user master data")
axs[1].grid(axis='y')
axs[1].set_xlabel('procedures code')
axs[1].set_xticklabels([])

master_data.icd_code_P.value_counts().to_pandas().head(30).plot.bar(ax=axs[2],
color='green')
axs[2].set_title("30 icd_code_P by user master data")
axs[2].grid(axis='y')

plt.tight_layout()
plt.show()

fig, axs = plt.subplots(3, 1, figsize=(20,10), sharey=True)

master_data.icd_code.value_counts().to_pandas().plot.bar(ax=axs[0])
axs[0].set_title("icd_code by user master data")
axs[0].grid(axis='y')
axs[0].set_xlabel('diagnose code')
axs[0].set_xticklabels([])
```

```
master_data.icd_code.value_counts().to_pandas().head(300).plot.bar(ax=axs[1])
axs[1].set_title("300 icd_code by user master data")
axs[1].grid(axis='y')
axs[1].set_xlabel('diagnose code ')
axs[1].set_xticklabels([])

master_data.icd_code.value_counts().to_pandas().head(30).plot.bar(ax=axs[2])
axs[2].set_title("30 icd_code by user master data")
axs[2].grid(axis='y')

plt.tight_layout()
plt.show()

import matplotlib.pyplot as plt

y = master_data['curr_service'].to_pandas()
plt.figure(figsize=(15,5))

plt.hist(y)
plt.xlabel('curr service')
plt.ylabel('Frequency')
plt.title('Distribution of curr service')
plt.tight_layout()
plt.show()

"""### (sampling only icd_code)"""

import cudf, cuml, cupy

import gc
gc.collect()

master_data.shape

master_data.icd_code.to_pandas().value_counts().shape

# Create a new column 'icd_code_num' that assigns a numerical value to each
unique 'icd_code' value
master_data['icd_code_num'] =
master_data['icd_code'].astype('category').cat.codes

# Count the number of occurrences of each unique numerical value in the
'icd_code_num' column
value_counts = master_data['icd_code_num'].value_counts()

# Define the number of bins
```

```
n_bins = 5000

# Select the `n_bins` most frequent unique numerical values
top_n_codes = value_counts.nlargest(n_bins).index

# Filter the original DataFrame to only include rows where the 'icd_code_num'
column is one of the selected values
master_data = master_data[master_data['icd_code_num'].isin(top_n_codes)]

# Group data by current service and randomly sample a certain number of rows
n_max = 40000
master_data = master_data.groupby('curr_service').apply(lambda x:
x.sample(n=min(n_max, len(x))))

# Reset the index and drop unnecessary columns
master_data  = master_data.reset_index(level=[0,1], drop=True)

# Group data by ICD code and randomly sample a certain number of rows
n_max = 1000
master_data = master_data.groupby('icd_code').apply(lambda x:
x.sample(n=min(n_max, len(x))))

# Reset the index and drop unnecessary columns
master_data = master_data.reset_index(level=[0,1], drop=True)

# Group data by current service and randomly sample a certain number of rows
n_max = 1000
master_data = master_data.groupby('icd_code_P').apply(lambda x:
x.sample(n=min(n_max, len(x))))

# Reset the index and drop unnecessary columns
sampled_data = master_data.reset_index(level=[0,1], drop=True)

del master_data

print(master_data.shape)
print(master_data.icd_code.value_counts().shape)
print(master_data.icd_code_P.value_counts().shape)
print(master_data.curr_service.value_counts().shape)
print(master_data.curr_service.shape)

print(sampled_data.shape)
print(sampled_data.icd_code.value_counts().shape)
print(sampled_data.icd_code_P.value_counts().shape)
print(sampled_data.curr_service.value_counts().shape)
print(sampled_data.curr_service.shape)
```

```
print(sampled_data.shape)
print(sampled_data.icd_code.value_counts().shape)
print(sampled_data.icd_code_P.value_counts().shape)
print(sampled_data.curr_service.value_counts().shape)
print(sampled_data.curr_service.shape)

sampled_data['icd_code_P_1']=sampled_data['icd_code_P']
sampled_data['subject_id_1']=sampled_data['subject_id']

#Convert icd_code_P_1 and subject_id_1 to category
sampled_data['icd_code_P_1'] = sampled_data['icd_code_P_1'].astype('category')
sampled_data['subject_id_1'] = sampled_data['subject_id_1'].astype('category')

#Assign new codes for icd_code_P_1 and subject_id_1
sampled_data['icd_code_P_1'] = sampled_data['icd_code_P_1'].cat.codes
sampled_data['subject_id_1'] = sampled_data['subject_id_1'].cat.codes

import matplotlib.pyplot as plt

fig, axs = plt.subplots(3, 1, figsize=(20,10), sharey=True)

sampled_data.icd_code_P.value_counts().to_pandas().head(9000).plot.bar(ax=axs[
0],color='green')
axs[0].set_title("icd_code_P by user after sampling")
axs[0].grid(axis='y')
axs[0].set_xlabel('procedures code')
axs[0].set_xticklabels([])

sampled_data.icd_code_P.value_counts().to_pandas().head(300).plot.bar(ax=axs[1
],color='green')
axs[1].set_title("300 icd_code_P by user after sampling")
axs[1].grid(axis='y')
axs[1].set_xlabel('procedures code')
axs[1].set_xticklabels([])

sampled_data.icd_code_P.value_counts().to_pandas().head(30).plot.bar(ax=axs[2]
,color='green')
axs[2].set_title("30 icd_code_P by user after sampling")
axs[2].grid(axis='y')

plt.tight_layout()
plt.show()

fig, axs = plt.subplots(3, 1, figsize=(20,10), sharey=True)

sampled_data.icd_code.value_counts().to_pandas().plot.bar(ax=axs[0])
axs[0].set_title("icd_code by user after sampling")
```

```
axs[0].grid(axis='y')
axs[0].set_xlabel('diagnose code')
axs[0].set_xticklabels([])

sampled_data.icd_code.value_counts().to_pandas().head(300).plot.bar(ax=axs[1])
axs[1].set_title("300 icd_code by user after sampling")
axs[1].grid(axis='y')
axs[1].set_xlabel('diagnose code')
axs[1].set_xticklabels([])

sampled_data.icd_code.value_counts().to_pandas().head(30).plot.bar(ax=axs[2])
axs[2].set_title("30 icd_code by user after sampling")
axs[2].grid(axis='y')

plt.tight_layout()
plt.show()

import matplotlib.pyplot as plt

y = sampled_data['curr_service'].to_pandas()
plt.figure(figsize=(15,5))

plt.hist(y)
plt.xlabel('y')
plt.ylabel('Frequency')
plt.title('Distribution of y')
plt.tight_layout()
plt.show()

sampled_data.to_parquet('/content/drive/MyDrive/123/rapids
jormungandr/sampled_data.parquet',compression='snappy')

chunk_size = 500000
for i in range(0, len(sampled_data), chunk_size):
    chunk = sampled_data.iloc[i:i+chunk_size]
    chunk.to_parquet('/content/drive/MyDrive/123/rapids jormungandr
(1)/sampled_data_part{}.parquet'.format(i), compression='snappy')

import pickle
#features
with open('/content/drive/MyDrive/123/rapids jormungandr/sampled_data.pickle',
'wb') as f:
    pickle.dump(sampled_data, f, protocol=pickle.HIGHEST_PROTOCOL)

import pickle
```

```
with open('/content/drive/MyDrive/123/rapids jormungandr/sampled_data.pickle',
'rb') as f:
    sampled_data = pickle.load(f)

"""##(prepare data)"""

import cudf, cuml, cupy, os

dataframe_list = []
for filename in os.listdir('/content/drive/MyDrive/123/rapids jormungandr
(1)/'):
    if filename.startswith('sampled_data_part'):
        path = '/content/drive/MyDrive/123/rapids jormungandr (1)/'+filename
        chunk = cudf.read_parquet(path)
        dataframe_list.append(chunk)

sampled_data = cudf.concat(dataframe_list)

sampled_data = cudf.read_parquet('/content/drive/MyDrive/123/rapids
jormungandr/sampled_data.parquet')
sampled_data

# procedures dataframe
procedures =
sampled_data[["icd_code_P_1","icd_code_P","icd_version_P","long_title_P"]]

# convert icd_code_P column to string type
procedures['icd_code_P'] = procedures['icd_code_P'].astype('str')

# get list of unique characters
unique_chars =
set(''.join(procedures['icd_code_P'].dropna().to_arrow().to_pylist()))

# create new columns for each unique character
for char in unique_chars:
    procedures[char] = 0

# update new columns with count of occurrences for each character
for char in unique_chars:
    procedures[char] = procedures['icd_code_P'].str.count(char)

procedures.drop(columns = ['icd_code_P'], inplace=True)
print(procedures.shape)

# user dataframe
user = sampled_data[['subject_id_1', 'icd_code_P_1','seq_num_P']]
print(user.shape)
```

```
user = user.rename(columns={'seq_num_P': 'y'})

import gc
gc.collect()

# Convert the DataFrame to a cuDF Array
context = sampled_data[["insurance", "marital_status", "race", "gender",
"prev_service",
"curr_service","anchor_age","icd_code","icd_version","seq_num"]]
# One-hot encode the columns
context = cudf.get_dummies(context,['insurance', 'marital_status', 'race',
'gender', 'prev_service', 'curr_service','icd_code'])
# Print the resulting cuDF Array
print(context.shape)

print(procedures.shape)
print(user.shape)
print(context.shape)

# Extra data long_title_P icd_code_P_P
procedures1 = procedures
features = procedures1.drop(["long_title_P",'icd_code_P_1'], axis=1).columns
print(features)

context1 = context.columns
print(context1)

complete_df = cudf.concat([user, procedures1[features], context[context1]],
axis=1)
print(complete_df.shape)

"""## (explore data)"""

import matplotlib.pyplot as plt

y = complete_df['y'].to_pandas()

plt.hist(y)
plt.xlabel('y')
plt.ylabel('Frequency')
plt.title('Distribution of y')
plt.show()

# remove outlier
q1 = complete_df['y'].quantile(0.01)
q3 = complete_df['y'].quantile(0.99)
complete_df['y'] = complete_df['y'].clip(lower=q1, upper=q3)
```

```
import matplotlib.pyplot as plt

y1 = complete_df['y'].to_pandas()

plt.hist(y1)
plt.xlabel('y')
plt.ylabel('Frequency')
plt.title('Distribution of y after remove the outlier')
plt.show()

procedures

import seaborn as sns
#Assuming df is your current cuDF dataframe, compute the correlation
a = cudf.concat([sampled_data[['subject_id_1']], procedures[features]],
axis=1).to_pandas()
corr = a.corr()
fig, ax = plt.subplots(figsize=(20,5))
sns.heatmap(corr, cmap='coolwarm', annot=False, ax=ax).set_title("Products x
Features Correlation")
plt.show()

import seaborn as sns
#Assuming df is your current cuDF dataframe, compute the correlation
a = cudf.concat([sampled_data[['subject_id_1']], procedures[features]],
axis=1).to_pandas()
fig, ax = plt.subplots(figsize=(20,5))
sns.heatmap(a==0, vmin=0, vmax=1, cbar=False, ax=ax).set_title("Patients x
Features")
plt.show()

import matplotlib.pyplot as plt

# Set the number of rows and columns for the subplots
nrows = 18
ncols = 2

# Create the figure and axes for the subplots
fig, axs = plt.subplots(nrows, ncols, figsize=(20, 50))

# Flatten the axes array
axs = axs.ravel()

# Iterate through each feature column
for i, feature in enumerate(features):
    # Plot the histogram of the feature
```

```
    axs[i].hist(procedures1[feature].to_pandas())
    axs[i].set_title(feature)
    axs[i].set_xlabel(feature)
    axs[i].set_ylabel('Frequency')

# Tight layout and show the plot
plt.tight_layout()
plt.show()

import matplotlib.pyplot as plt

# Set the number of rows and columns for the subplots
nrows = 37
ncols = 2

# Create the figure and axes for the subplots
fig, axs = plt.subplots(nrows, ncols, figsize=(20, 60))

# Flatten the axes array
axs = axs.ravel()

#define age
seq_num = context['seq_num'].unique
ages = context['anchor_age'].to_pandas().unique().tolist()
ages.sort()

# Iterate through each seq_num column
for i, age in enumerate(ages):
    # Plot the histogram of the feature
    axs[i].hist(context['seq_num'].to_pandas())
    axs[i].set_title(age)
    axs[i].set_xlabel('seq_num')
    axs[i].set_ylabel('Frequency')

# Tight layout and show the plot
plt.tight_layout()
plt.show()

import matplotlib.pyplot as plt

# Set the number of rows and columns for the subplots
nrows = 37
ncols = 2

# Create the figure and axes for the subplots
fig, axs = plt.subplots(nrows, ncols, figsize=(20, 60))
```

```
# Flatten the axes array
axs = axs.ravel()

#define sequence numbers
seq_nums = context['seq_num'].unique()

# Iterate through each seq_num column
for i, seq_num in enumerate(seq_nums):
    # Filter the context dataframe to only include rows with the current
sequence number
    seq_num_context = context[context['seq_num'] == seq_num]

    # Plot the histogram of the anchor_age feature for the current sequence
number
    axs[i].hist(seq_num_context['anchor_age'].to_pandas())
    axs[i].set_title(f'Sequence Number {seq_num}')
    axs[i].set_xlabel('anchor_age')
    axs[i].set_ylabel('Frequency')

# Tight layout and show the plot
plt.tight_layout()
plt.show()

from sklearn.preprocessing import StandardScaler

# Create a list of columns to standardize
numerical_columns = complete_df.select_dtypes(include=['float32',
'int32']).columns
columns_to_standardize = [col for col in numerical_columns if col not in
['subject_id_1', 'icd_code_P_1', 'icd_version_P']]

# Initialize the StandardScaler object
scaler = StandardScaler()

# Fit and transform the data
complete_df[columns_to_standardize] =
scaler.fit_transform(complete_df[columns_to_standardize].to_pandas())

print(complete_df.shape)

"""## ( training the model )"""

from cuml.model_selection import train_test_split

# split dataframe
train_df, test_df = train_test_split(complete_df, train_size=0.8)
```

```
print(train_df.shape)

train_df

import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from scipy import stats

## for plotting
import matplotlib.pyplot as plt
import seaborn as sns

## for machine learning
from sklearn import metrics, preprocessing

## for deep learning
from tensorflow.keras import models, layers, utils  #(2.6.0)

!nvidia-smi



embeddings_size = 50
usr, prd = user.shape[0], procedures1.shape[0]
feat = len(features)
ctx = len(context1)

############################################ COLLABORATIVE FILTERING
#####################################################
# Input layer
xusers_in = layers.Input(name="xusers_in", shape=(1,))
xproducts_in = layers.Input(name="xproducts_in", shape=(1,))

# A) Matrix Factorization
## embeddings and reshape
cf_xusers_emb = layers.Embedding(name="cf_xusers_emb", input_dim=usr,
output_dim=embeddings_size)(xusers_in)
cf_xusers = layers.Reshape(name='cf_xusers',
target_shape=(embeddings_size,))(cf_xusers_emb)
## embeddings and reshape
cf_xproducts_emb = layers.Embedding(name="cf_xproducts_emb", input_dim=prd,
output_dim=embeddings_size)(xproducts_in)
cf_xproducts = layers.Reshape(name='cf_xproducts',
target_shape=(embeddings_size,))(cf_xproducts_emb)
## product
```

```
cf_xx = layers.Dot(name='cf_xx', normalize=True, axes=1)([cf_xusers,
cf_xproducts])

# B) Neural Network
## embeddings and reshape
nn_xusers_emb = layers.Embedding(name="nn_xusers_emb", input_dim=usr,
output_dim=embeddings_size)(xusers_in)
nn_xusers = layers.Reshape(name='nn_xusers',
target_shape=(embeddings_size,))(nn_xusers_emb)
## embeddings and reshape
nn_xproducts_emb = layers.Embedding(name="nn_xproducts_emb", input_dim=prd,
output_dim=embeddings_size)(xproducts_in)
nn_xproducts = layers.Reshape(name='nn_xproducts',
target_shape=(embeddings_size,))(nn_xproducts_emb)
## concat and dense
nn_xx = layers.Concatenate()([nn_xusers, nn_xproducts])
nn_xx = layers.Dense(name="nn_xx", units=int(embeddings_size/2),
activation='relu')(nn_xx)

############################################ CONTENT BASED
#####################################################
# Product Features
features_in = layers.Input(name="features_in", shape=(feat,))
features_x = layers.Dense(name="features_x", units=feat,
activation='relu')(features_in)

############################################ KNOWLEDGE BASED
#####################################################
# Context
contexts_in = layers.Input(name="contexts_in", shape=(ctx,))
context_x = layers.Dense(name="context_x", units=ctx,
activation='relu')(contexts_in)

############################################ OUTPUT
#####################################################
# Merge all
y_out = layers.Concatenate()([cf_xx, nn_xx, features_x, context_x])
y_out = layers.Dense(name="y_out", units=1, activation='linear')(y_out)

# Compile
model = models.Model(inputs=[xusers_in,xproducts_in, features_in,
contexts_in], outputs=y_out, name="Hybrid_Model")
model.compile(optimizer='adam', loss='mean_absolute_error',
metrics=['mean_absolute_percentage_error'])
model.summary()

gc.collect()
```

```
train_df_1 = train_df["subject_id_1"].to_numpy()
train_df_2 = train_df["icd_code_P_1"].to_numpy()
train_df_3 = train_df[features].to_numpy()
train_df_4 = train_df[context1].to_numpy()
train_df_y = train_df["y"].to_numpy()

# Train the model using the TensorFlow tensors as input
training = model.fit(x=[train_df_1, train_df_2, train_df_3, train_df_4],
y=train_df_y, epochs=120,batch_size=300,  shuffle=True, validation_split=0.15,
verbose=1)

import gc
gc.collect()

#plt.plot(training.history['mean_absolute_percentage_error'])
plt.plot(training.history['val_mean_absolute_percentage_error'])
plt.title('model val_mean_absolute_percentage_error')
plt.xlabel('epoch')
plt.ylabel('mean_absolute_percentage_error')
plt.legend(['train','validate'], loc ='upper left')
plt.show()

plt.plot(training.history['mean_absolute_percentage_error'])
plt.plot(training.history['val_mean_absolute_percentage_error'])
plt.title('model mean_absolute_percentage_error')
plt.xlabel('epoch')
plt.ylabel('mean_absolute_percentage_error')
plt.legend(['train','validate'], loc ='upper left')
plt.show()

plt.plot(training.history['loss'])
plt.plot(training.history['val_loss'])
plt.title('model loss')
plt.xlabel('epoch')
plt.ylabel('loss')
plt.legend(['train','validate'], loc ='upper left')
plt.show()

#plt.plot(training.history['loss'])
plt.plot(training.history['val_loss'])
plt.title('model val_loss')
plt.xlabel('epoch')
plt.ylabel('loss')
plt.legend(['train','validate'], loc ='upper left')
plt.show()
```

```
"""##(test the model)"""

test_df

test_df1 = test_df.to_pandas()

# test
test_df1["yhat"] = model.predict([test_df1["subject_id_1"],
test_df1["icd_code_P_1"], test_df1[features], test_df1[context1]])
test_df1

# overall
y_test = test_df1["y"]
predicted = test_df1["yhat"]

print("Mean Absolute Error (Σ|y-pred|/n):",
"{:,.0f}".format(metrics.mean_absolute_error(y_test, predicted)))
print("Mean Absolute Perc Error (Σ(|y-pred|/y)/n):",
str(round(np.mean(np.abs((y_test-predicted)/predicted)), 2))+"%")

# Evaluate
def mean_reciprocal_rank(y_test, predicted):
    score = []
    for product in y_test:
        mrr = 1 / (list(predicted).index(product) + 1) if product in predicted
else 0
        score.append(mrr)
    return np.mean(score)

# test all
sampled_data1 = sampled_data.to_pandas()
hasil =
test_df1.merge(sampled_data1[["long_title_P","icd_code_P","icd_version_P"]],
                      left_on="icd_code_P_1",
right_index=True).sort_values("yhat", ascending=False)

hasil

"""#( production generate top 10 procedures )"""

from IPython.display import display
import cudf, cuml
import gc
import pandas as pd
import pickle
# Import the necessary libraries
from tensorflow.keras.models import load_model
```

```
# Load the model from the file
model = load_model('/content/drive/MyDrive/123/rapids jormungandr/januari 25
januari 2023/Salinan jormungandr.h5')

# Load sampled data
#sampled_data = cudf.read_parquet('/content/drive/MyDrive/123/rapids
jormungandr/januari 25 januari 2023/Salinan sampled_data.pickle')
#sampled_data = sampled_data.to_pandas()

with open('/content/drive/MyDrive/123/rapids jormungandr/januari 25 januari
2023/Salinan sampled_data.pickle', 'rb') as f:
    sampled_data = pickle.load(f)
with open('/content/drive/MyDrive/123/rapids jormungandr/januari 25 januari
2023/Salinan complete_df.pickle', 'rb') as f:
    complete_df = pickle.load(f)
with open('/content/drive/MyDrive/123/rapids jormungandr/januari 25 januari
2023/Salinan features.pickle', 'rb') as f:
    features = pickle.load(f)
with open('/content/drive/MyDrive/123/rapids jormungandr/januari 25 januari
2023/Salinan context1.pickle', 'rb') as f:
    context1 = pickle.load(f)

#complete_df = complete_df.to_pandas()

sampled_data

complete_df

# Load sampled data
procedures = cudf.read_parquet('/content/drive/MyDrive/123/rapids
jormungandr/januari 25 januari 2023/Salinan procedures.parquet')
#procedures1 = procedures.to_pandas()
procedures1 =
procedures.drop_duplicates(subset='icd_code_P_1').reset_index(drop=True)
procedures1

from pandas.io.formats.style import Subset
def get_user_context(user_id, complete_df, context1):
    context1 = [col for col in context1 if col in complete_df.columns]
    return complete_df.loc[complete_df["subject_id_1"]==user_id,
["subject_id_1"] + context1]

def create_procedures_df(user_id, procedures1):
    procedures1 = procedures1.drop_duplicates().reset_index(drop=True)
    procedures1['subject_id_1'] = user_id
    return procedures1
```

```
def generate_recommendations(user_id, model, procedures1, complete_df,
features, context1):
    user_context = get_user_context(user_id, complete_df, context1)
    procedures1 = create_procedures_df(user_id, procedures1)
    procedures1 = cudf.merge(procedures1, user_context, on='subject_id_1',
how='left')
    procedures1 = procedures1.to_pandas()

    procedures1["yhat"] = model.predict([procedures1["subject_id_1"],
procedures1["icd_code_P_1"], procedures1[features], procedures1[context1]])
    return procedures1.sort_values("yhat",
ascending=False).drop_duplicates(subset='icd_code_P_1').nlargest(10, 'yhat')

# Choose a random user
user_id = 50
user_data = sampled_data.loc[sampled_data['subject_id_1'] == user_id]

# Generate recommendations for the user
rekomendasiX6 = generate_recommendations(user_id, model, procedures1,
complete_df, features, context1)

display(user_data[['long_title','seq_num','icd_code','icd_version']])
print("Hey, it seems you have been diagnosed with:")
print(user_data['long_title'].unique().to_pandas().squeeze())
print("and also have procedures records for:")
print(user_data['long_title_P'].unique().to_pandas().squeeze())
print("We have some interesting procedures that may be of interest to you.")
display(rekomendasiX6[['long_title_P', 'yhat']])

"""# Rapids"""

# This get the RAPIDS-Colab install files and test check your GPU.  Run this
and the next cell only.
# Please read the output of this cell.  If your Colab Instance is not RAPIDS
compatible, it will warn you and give you remediation steps.
!git clone https://github.com/rapidsai/rapidsai-csp-utils.git
!python rapidsai-csp-utils/colab/env-check.py

# This will update the Colab environment and restart the kernel.  Don't run
the next cell until you see the session crash.
!bash rapidsai-csp-utils/colab/update_gcc.sh
import os
os._exit(00)
```

```
# This will install CondaColab.  This will restart your kernel one last
time.  Run this cell by itself and only run the next cell once you see the
session crash.
import condacolab
condacolab.install()

# you can now run the rest of the cells as normal
import condacolab
condacolab.check()

# Installing RAPIDS is now 'python rapidsai-csp-utils/colab/install_rapids.py
<release> <packages>'
# The <release> options are 'stable' and 'nightly'.  Leaving it blank or
adding any other words will default to stable.
!python rapidsai-csp-utils/colab/install_rapids.py stable
import os
os.environ['NUMBAPRO_NVVM'] = '/usr/local/cuda/nvvm/lib64/libnvvm.so'
os.environ['NUMBAPRO_LIBDEVICE'] = '/usr/local/cuda/nvvm/libdevice/'
os.environ['CONDA_PREFIX'] = '/usr/local'
#!pip uninstall cupy -y

import torch
torch.cuda.is_available()
```